FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

TRABAJO FIN DE CARRERA

Uburyo Volume I: Designing of a Sustainable System of Loans for Education

AUTOR: EDUARDO MARTINEZ-LARRAZ SOLIS
COAUTOR: MÁXIMO RAMÍREZ ROBLES
TUTOR: SUSANA MUÑOZ HERNANDEZ

November 2010

A mis padres por su paciencia infinita. A mis hermanos y hermana por darle a todo la vuelta. A Rubén, Raquel, Jorge, Goñi y demás compinches, que me hacen plantearme las cosas desde otro punto de vista. A Máximo por ser más que compañero en esta aventura. A TEDECO por la confianza depositada. A la UNG por la acogida y participación en el proyecto. A la comunidad de SW Libre por la inspiración y todo ese código disponible ;-). Y, por supuesto, a Elisa, por apoyarme siempre, por ponerme los pies en la tierra y por acompañarme hasta Burundi, hasta el infinito y más allá.

Contents

List of Figures

List of Tables

Nomenclature

ANSI/IEEE American National Standards Institute/Institute of Electrical and Electronics Engineers
API Application Programming Interface
CVS Concurrent Version System
DAO Data Access Objects
FBU Burundi franc
FI Computer Science and Engineering Faculty
FK Foreign Key
GPL General Public License
GUI Graphic User Interface
HTTP HyperText Transfer Protocol
ICT Information and Communication Technologies
IDE Integrated Development Enviroment
LFA Logical Framework Approach
MVC Model-View-Controller
NGO Non-Governmental Organization
PK Primary Key
SMS Short Message Service
TEDECO TEcnology for DEvelopment and COoperation
TESON Technology for Sustainability at University of Ngozi
TICAMEN Information and Communication Technologies Applied to the Improvement of Education in Ngozi
UML Unified Modeling Language
UNG University of Ngozi
UPM Technical University of Madrid

Part 1

FIRST PART: DESIGNING

CHAPTER 1

INTRODUCTION, MOTIVATION

1.1. Analysis of the problem

This End of Degree Project belongs to the project Information and Communication Technologies Applied to the Improvement of Education in Ngozi[TED08] (code 07-CAP2-0487), from the cooperation group TEDECO and founded by the Technical University of Madrid.

1.1.1. TEDECO.

TEDECO (TEchnology for DEvelopment and COoperation) is a cooperation for development group from Technical University of Madrid (*UPM* from the Spanish abbreviation, *Universidad Politécnica de Madrid*).

In January 2006 at the UPM was received an urgent call from a university from Burundi,specifically the University of Ngozi (UNG): "We urgently need the help of cooperating teachers in a University of Burundi." Responded to this call two professors from the Computer Science and Engineering Faculty (FI by its Spanish acronym, *Facultad de Informática*) that during the months of February, March and April 2006 made both stays at the University of Ngozi.

After this experience decided to focus aid on a global way and so TEDECO Group was created as a UPM's cooperation group.

How its name indicates, TEDECO is a specialized group in Information and Communication Technologies (ICT) whose goal is to use New Technologies to provide centers (specially educational centers) in developing countries the infrastructure needed to develop and improve their situation.

1.1.1.1. *TESON.*

When TEDECO analyzed the situation of UNG at 2006 discovered that it was in a precarious situation. University was submerged in a lot of technical, logistic, educational and administrative problems that seriously threatened its existence.

Project Technology for Sustainability at University of Ngozi[TED06] (TESON from the Spanish abbreviation: *TEcnología para la SOstenibilidad de la Universidad de Ngozi*) was created in order to assist in technological development of an educational center, focusing on University of Ngozi by: delivering technical training to university staff and also students, adapting the Computer Center and, especially, giving to the university an Internet access. In parallel, University of Salamanca was carrying out a project for introducing electricity based on solar panels with batteries.

FIGURE 1.1. Logo of TEDECO

The specific aim of TESON project is University of Ngozi development, allowing an improvement in student's training with parallel development of the city. The main project's goal, focusing on University of Ngozi context, is the enrichment of its Computer Center as a main work tool and technical development of their students. To achieve this objective is required the installation of an autonomous power management, that will avoid the lack of electricity, caused by continual cuts in existing facilities. Chosen alternative identifies that next target is the installation of an energy production system based on solar cells. Another goal is to achieve the installation of an Internet access system via satellite. Installation cost is not high and its maintenance can be funded through the creation of a cyber. Finally, TEDECO considers an indispensable contribution, the delivery of educational courses to both students and staff responsible for classrooms and Computer Center maintenance, in order to provide enough training for a good provided material management and administration.

This project ended in 2007 with very good results.

1.1.1.2. *TICAMEN.*

Information and Communication Technologies Applied to the Improvement of Education in Ngozi[TED08] (TICAMEN from the Spanish abbreviation: *Tecnologías de la Información y la Comunicación Aplicadas a la Mejora de la Enseñanza en Ngozi)* project puts the target in strengthen information systems infrastructure and communications at University of Ngozi to facilitate teaching, administration, accounting and scheduling services. This will ensure a quality education appropriated to local people needs and financially sustainable.

TICAMEN project is part of the work line that the group was following in recent years. It attempts to find using ICT, an useful tool for institutions in developing countries, that can overcome typical problems, achieve sustainability and improve their services. This is possible because it has been carried out, during the last years, a hardware preparation that was needed to successfully complete a pilot experience in an institution of a developing country.

This project aims to develop a series of free software tools to use them as a first pilot destination at the University of Ngozi. These tools include features related the institution activity:

- Secretary of students
- Courses and teachers
- Financial management, material inventory, payroll and procurement management of temporary teachers
- Library
- Cyber Management
- Management of a scholarship scheme.

Every tool is intended to be reused in similar institutions. And, as well as its development, is part of this project requirements analysis with institution agreement, local staff training and testing period for future treatment and improvement. In parallel toe threalization of these fundamental goals, monitoring the proper maintenance of hardware systems will be an additional task, as well as new tools support training to local system maintainers in order to ensure system sustainability.

1.1.2. Problems of the educational centers in precarious situation in developing countries.

Most of the problems of these educational centers have been collected in the following schema (figure 1.2). It is a detected problems tree that resume the similar trees created

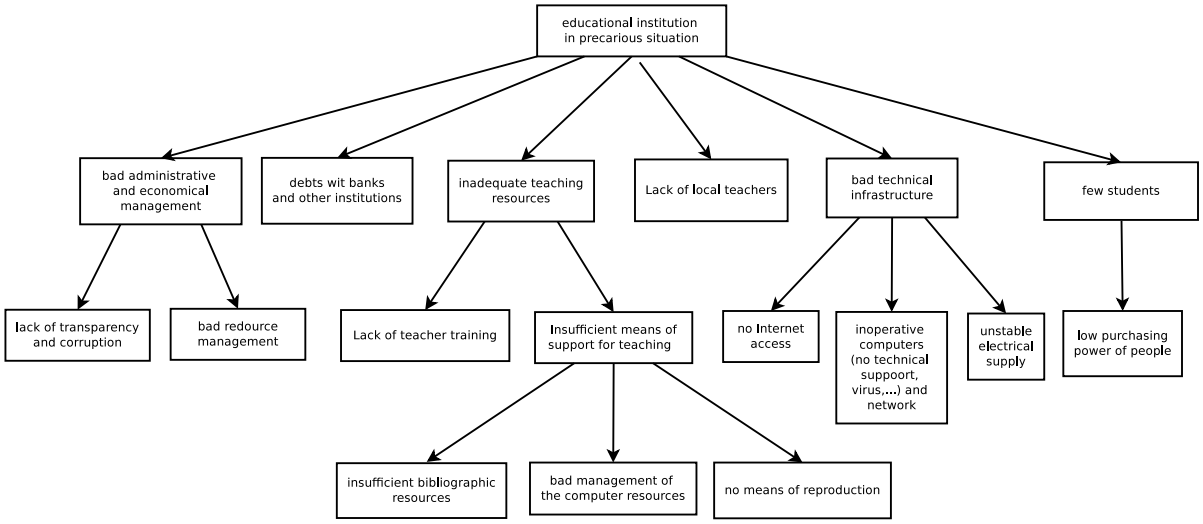


FIGURE 1.2. Founded problems tree

in the documentation of the two projects[TED06, TED08]. These trees were created following the Logical Framework Approach (LFA or EML from its Spanish abbreviations *Enfoque de Marco Lógico*)[NOR08]. The LFA is a management tool mainly used in the design, monitoring and evaluation of international development projects[WIK09b].

1.2. Problems of the conventional systems of scholarship

"A scholarship is an award of financial aid for a student to further education. Scholarships are awarded on various criteria usually reflecting the values and purposes of the donor or founder of the award" [WIK10h]. In this way we have the following classification:

- Merit-based: These awards are based on a student's athletic, academic, artistic or other abilities, and often factor in an applicant's community service record and extracurricular activities.
- Need-based: These awards are based on the student and family's financial record.
- Student-specific: These are scholarships where applicants must initially qualify by race, gender, religion, family and medical history, or many other student-specific factors.
- Career-specific: These are scholarships awarded by a college or university to students planning to pursue a specific field of study.

Traditionally a scholarship system has the scheme showed in the figure 1.3.
One organization give scholarships to students, and they can pay their studies.
But this system has some associated problems:

(1) Students cannot properly appreciate the value of scholarships. Students only need to ask for a scholarship and, if they are awarded, they do not need to do anything else. If they do not study there is no problem because they are only a beneficiary. So, when awarded students leave their studies, the related scholarships are lost.

(2) This system is not sustainable along the time. The system needs a constant flow of money that the organization provides to students. If the organization disappears, the scholarship system disappears too.

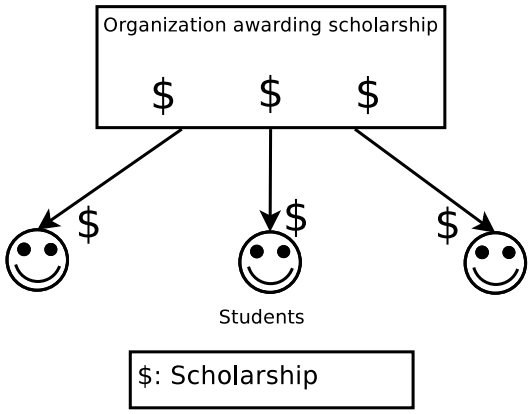


FIGURE 1.3. Diagram of a traditional scholarship system

(3) Corruption in the system. This kind of systems manage a very big quantity of money. It is very common that not all the money invested in the system will be used by students. Other kind of corruption can appear during the assignment process. In this case, aid are awarded unfairly.

And these three problems grow when this system is applied in a developing countries:

- In many places, cooperation is understood as simply charity. Thus, during too many years has been generated in these places a sense of mendicancy. This sense produces that people prefer to ask for a solution, becoming dependent, instead of working in a solution. They simply ask for scholarship because this is a free service that they may easily get. Therefore, the percentage of students who leave their studies grows and money invested in their education is lost.
- Most of the scholarships systems are maintained by Non-Governmental Organizations (NGOs) or other type of external organizations. Usually, these organizations have different goals and divide their resources according to their priorities. If the organization disappears or it needs to move their resources to other highest priority initiative, the system disappears too.
- Corruption is one of the most important problems in developing countries, specially for economic and social development. Besides, corruption is widespread and part of everyday life. Society has learned to live with it, even considering it, fatalistically, as an integral part of their culture. With this situation is very common that money disappears or scholarships are awarded without following a fair criteria.

In some countries (like Nordic countries, Canada, United Kingdom, Belgium and Germany) is used another kind of scholarships system: the scholarships-loans system. The scheme of this system is represented in the figure 1.4.

An organization, sometimes the own university or the government, gives scholarships to pay student's courses (in particular tuition fees). When students finish their studies and start to work in a company, they must return loaned money to the organization. To assure the money return to the organization, this system is backed by some contracts.

With this scheme, the organization tries to add sustainability and responsibility to traditional systems. Sustainability, because students return the money and organizations

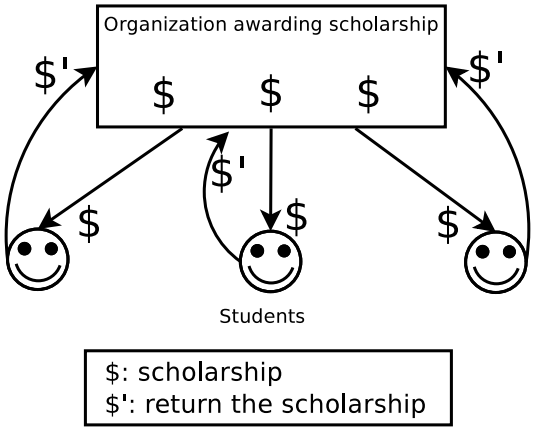


FIGURE 1.4. Diagram of a scholarships-loans system

may give it to other students. Thus, it is not necessary an external and continuous financing. Responsibility, because students sign contracts which commit them to give back their loans. In developed countries, this experience works evidenced by the fact that is being implemented in Europe through Bologna plan.

But in developing countries the social inequalities introduce a new problem with this model. Difference between university registration fees and normal salaries is very high and this makes very difficult to students to return the loan. Moreover, in case of getting that money is probably that they might have other more important and urgent expenses, such as food or family health. If students ca not return the loan, all the benefits disappear.

1.3. Proposal

As we have seen, the project TICAMEN detected the need to create a new scholarships system. But this system has to cope with problems seen in the previous point.

In collaboration with UNG, the old system of scholarships was studied and compared with known modern models. As a result of this study is designed the *Sustainable System of Loans* to study: a new scholarships system whose main characteristics make it a particular suitable system for underdeveloped countries.

One of the first conclusions reached is that the system should be transparent. The new system should prevent all possible ways of fraud, from stealing the money for the scholarships to the unfairly election of beneficiaries of aid. Thus, the system must ensure democratic functioning of aids allocation, taking into account needs of all students who request them. It must ensure that different processes are always supervised automatically and also by a commission, with external and internal members of the institution, who will take the most controversial decisions during the process. Among other tasks, they will be those who, through a clean, clear and transparent process, say to whom aid is assigned.

Other main characteristic is sustainability. All modern education plans (like Bologne plan in Europe) are implanting scholarships-loans systems because they only need to put an initial quantity of money to run the system. With this model the system can be sustainable if the student may return the money. In this case, due to problems seen in the last section, the system need to include new ways to return the money, mainly with works (figure 1.5). This will be explained in the next section.

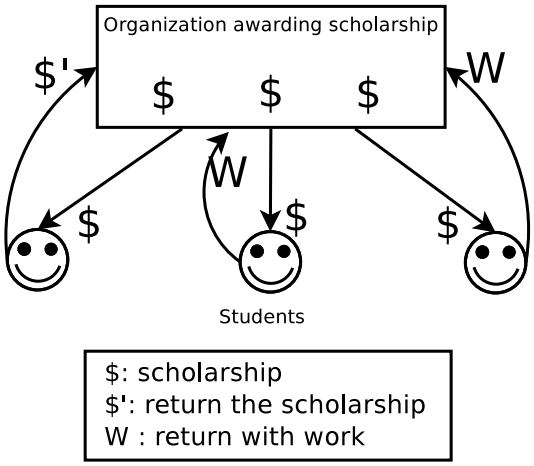


FIGURE 1.5. Diagram of the new scholarships-loans system

TABLE 1.1. Summary of possible ways to return money back

Return mode	Characteristics	Money returned	Money saved	Money earned	Other considerations
money	Students return their loans in cash	YES	NO	NO	
working for the institution	Students do some work for the institution	NO	YES	NO	Gain “added value”
job from the institution	Students do some work from the institution to other institutions or clients	YES	NO	YES	
job at an external company	Students do some work in an external company	YES	NO	YES	

Added to sustainability this kind of systems create a responsibility sense to students. They are not only beneficiaries of the system. They are a very important piece in the system because if they do not undertakes to repay the money the system fails. It is very important to promote this responsibility so the new system may have explicit mechanisms to achieve it. Some mechanisms could be an information campaign about how the system is backed on contracts.

The last main characteristic is that the system will be supported by software. At first glance may seem a forced conclusion because TEDECO consists mainly of computer scientist, making good the saying “for a hammer, everything is a nail”. But in this case is very important the use of software to ensure transparency during the process. If the system is managed by a program, it is very easy to keep all the movements and actions during the whole process and who do them. And this kind of watching is a very good way to fight against corruption.

1.3.1. The money returning system.

At this point is the time to explain different levels and effects that introduce this proposal in the system. They are summarized in the table 1.1.

First of all, students can return the quantity of the loan in cash. Although it is a way that will be not normally used in developing countries context, it was preferred to keep this possibility. All money will be returned to the bag of money so that the loan will be paid.

Secondly, students can do some tasks for the educational center or institution. For example, students of computer science could do maintenance of computers, any student could help at the library or paint a wall. There are particular tasks, for each institution, that add value to it at the same time that allow it to save some money. With this *added value,* the institution gains prestige because it is able to provide more services and better installations, attracting more students in future. At this point it is very difficult that all money can be recovered, but the global institution is strengthen with more students. Note that money discount to student debts should be equal to or greater than savings from the job. It is the way for avoiding to start a situation of exploitation of students.

The third returning is when students are able to be teachers or assistants in courses offered by the institution. Educational centers of developing countries, specially universities, are the knowledge and innovation centers. It is very common that these institutions organize some courses for different kind of professionals that pay for them. There exists the possibility for students with loans from these institutions of being teachers or assistants in this kind of courses. Thus, the institution gains money from the courses and reduces student debts. Entering the share of teacher's salary in the bag of money with which the loans are paid, sustainability is got in this kind of jobs.

Fourth and last mode for returning the money is very common in developed countries: practices in external companies. Students of the last years, or with the studies finished but with an outstanding debt, can do some jobs in external companies and these companies pay to the institution. This payment should be equal to or greater than quantity discounted to student debts. It is the way to do not start a situation of exploitation. While reducing student debt, introduce the money in the bag of money with which the loans are paid.

1.4. Objectives

The main goal of this work is to study, design and develop processes and software that supports the managing of the Sustainable System of Loans. They are intended to alleviate some of the problems shown in the Founded Problems Tree (figure 1.2), mainly:

- Lack of transparency and corruption: monitoring all the process with a transparent software and using a mixed commission to take the important decisions in the process.
- Low purchasing power of people: offering the opportunity to study students without resources

Following with these last point, the project is called *Uburyo*, a Kirundi language word that means *Opportunity.*

Besides this main objective, there are a second objective that is to do a deployment of Uburyo in the University of Ngozi.

1.5. Technologies

To meet the goals shown in the last section we need to take decisions about which appropriate technologies we will develope and use.

1.5.1. Appropriate technology.

"Appropriate technology is technology designed with special consideration to the environmental, ethical, cultural, social, political, and economical aspects of the community it is intended for. With these goals in mind, appropriate technology proponents claim their methods require fewer resources, are easier to maintain, and have less of an impact on the environment compared to techniques from mainstream technology, which they contend is wasteful and environmentally polluting". [WIK10a].

Since technology is an essential factor of production, the introduction of new technologies or development of existing in a society is one of the means to contribute to its development. Hence the importance of technical cooperation in its many variants, as part of development cooperation, is included by NGOs in productive projects.

This approach to appropriate technology (emerged in response to the limitations of traditional technologies and the problems arising from the transfer modern technology to poor countries) is sophisticated and capital intensive.

Traditional technologies are often highly adapted to environmental conditions, economic and social place, thanks to that have been developed and used for long periods of time. They also tend to use local materials, which facilitates the maintenance and repair of equipment. However, there is often a technology that, in a context of economic changes on the national or international market, offering an inadequate production and income.

Therefore, cooperation in technological areas can bring benefits to small farmers and other poor sectors, such as increasing revenue or saving time. However, this area of international aid has often been one that presents more problems, errors and losses. These include the dependence created regarding expensive inputs and spare parts to be imported, the resulting inability of people to use and maintain the equipment due to lack of money or knowledge, environmental damage, or increased social and gender differences. These problems are often due to technology is considered in error as a "neutral" factor, which simply helps solve problems, so it is not carried out the necessary analysis of the impact that its introduction will have on the economic, cultural or social level. So, often overlooked that the new technologies, depending on who controls and revenue they provide, will make changes in gender relations (women often do not exercise such control), social relations, the division of labor, access to natural resources, etc. In short, a useful technology in one context may not be in another.

The appropriate technology approach appears in the 70s as an alternative to the concept of transfer of modern technology and its problems, and is the type of technology most commonly used by NGOs in their development and poverty alleviation projects. The concept was formulated by E. F. Schumacher in his book *Small is Beautiful*[SCH73], on the basis of Gandhi's ideas on the use of small scale technologies that would improve the standard of living of the rural population in India. Schumacher, thinking more development oriented people to obtain benefits, proposed and defined appropriate technology as a simple technology, *small scale, low cost and non-violent* [PZ05].

1.5.2. Free Software.

Within the software world there is a branch to the appropriate technology we can call its own right: Free Software.

The Free Software Foundation proposes the following definition for Free Software:

- Free software is a matter of the user's freedom to run, copy, distribute, study, change and improve the software. More precisely, it means that the program's users have the four essential freedoms: [GNU10a]
  - The freedom to run the program, for any purpose (freedom 0).

- The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Then we are going to study this four freedoms and see how they convert Free Software in appropriate technology.

The freedom to run the program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job and purpose, without being required to communicate about it with the developer or any other specific entity. In this freedom, it is the user's purpose that matters. This freedom corresponds to the definition of appropriate technology because it allows the user to use this according to their own purposes. In this way the technology can be provided with new uses for which had not been designed at first time.

The freedom to redistribute copies must includes binary or executable forms of the program, as well as source code, for both modified and unmodified versions. This freedom is necessary for the freedoms 1 and 3, and contribute to the moral aspects of appropriate technology. Also, how all the people can see the source code, the vulnerabilities of the programs are solved more quickly, doing hardly programs that have less problems.

Freedom 1 includes the freedom to use your changed version in place of the original. This, combined with the freedom 2, allows to adapt technology to the special needs of a particular case, so as to make a good maintenance.

Freedom 3 includes the freedom to release your modified versions as free software. A free license may also permits other ways of releasing them; in other words, it does not have to be a copyleft license. As the freedom 2, this freedom contribute to the moral aspects of appropriate technology, doing that more people could use this technology.

Also these considerations it is necessary to add other characteristic to the Free Software: the low cost. Much has been written about the cost of free software because the Spanish translation often leads to misunderstandings. But the reality is that 99% of Free Software has no-cost for the final user.

All of this, added that Free Software normally no need a lot of resource to be executed, convert Free Software in appropriate technology. And is for this that we decide to develop our application using Free Software tools and to license the resultant software as Free Software (and the documentation with similar license).

1.5.3. Methodology and architecture.

Before to starting the development of the Uburyo Software, we need to take some decisions about the methodology and architecture to follow during the development process.

The main architecture will be similar to the most commons applications in Internet: a database with a web-based interface. As there were no special requirements on the database, it was decided to use a relational database.

1.5.4. Used tools.

Finally, we had to choose a set of tools to do the work, both code development and all documentation surrounding a project of this nature. All of them are Free Software for the reasons stated in section 1.5.2, and we are going to see why they were chosen:

- LyX[LYX10]: is a document processor based on LaTeX[LaT08] that encourages an approach to writing based on the structure of your documents so it can be used to edit large documents (books) or rigorous format (theses, articles for scientific journals) easily. In this project LyX was used to write the report.
- Dia[GNO10]: is a diagram creation program based in the GTK+ library project[GTK09]. In this project was used to draw all the own diagrams.
- Open Office[ORA10b]: is the leading open-source office software suite for word processing, spreadsheets, presentations, graphics, databases and more. In this project was used to write some manuals and presentations.
- Ubuntu[UBU10]: is a computer operating system based on the Debian GNU/Linux distribution. Although the Uburyo software can run on the most common systems (MS Windows, GNU/Linux based systems and other UNIX systems) , we use Ubuntu for development and testing.
- MySQL[ORA10a]: is a relational database manager, multi-thread and multi-user. In this project was used to manage the relational database that support the project's "heart". In addition to this database manager, was used MySQL Workbench[MyS10], a visual database design tool that was used to design the database and draw its schemes.
- Apache[APA10]: is a HTTP web server that run over multiple platforms. In this project was used to run an tested the application.
- Eclipse[ECL10]: is a multi-language software development environment comprising an Integrated Development Environment (IDE) and an extensible plug-in system. In this project was used to write the PHP code and facilitate the use of Concurrent Version System (CVS).
- Sourceforge[SOU10a]: is a web-based source code repository. It acts as a centralized location for software developers to control and manage open source software development. In this project was used as official repository, especially because has many tools integrated like CVS, BBForums, Manthis, mediaWiki,...
- CVS[CVS10]: The Concurrent Versions System is a free software revision control system in the field of software development. In this project was used to keep track of all work and all changes in a set of files, and allows several developers (potentially widely separated in space and/or time) to collaborate in the future. The CVS server is installed in the Sourceforge site and the client is inside Eclipse.
- MediaWiki[MED10]: is a popular free web-based wiki software application. Sourceforge allow to install easily a wiki based in this software, added to the official project site. In this project was used to maintain a collaborative site to generate documentation about the project.
- MantisBT[MAN10]: Mantis Bug Tracker is a free popular web-based bug-tracking system, offered integrate by Sourceforge in the project site. In this project was used to managed the bug that occurred during the installation and use of the first version.
- phpBB[PHP10]: is a free flat-forum bulletin board software solution that can be used to stay in touch with a group of people or can power your entire website. Sourceforge offers an easily installation of phpBB to create forums added to the official project site. In this project was used to create a community around the project that can discuss about it.

1.6. Document organization

This section is giving a brief overview about this document in order to make its reading easier. This report is the second volume of a work performed by two computer science students as their final studies project: Máximo Ramírez Robles and Eduardo Martinez-Larraz. Section 7.3 explains how different tasks where divided. Moreover, is highly recommended to read the second volume: Uburyo Volume II: Delivering of a sustainable system of loans for education [RAM10] for a complete understanding about the whole process.

This volume id divided into two different parts: The first part, in which is included this section itself, shows us all the designing process, from idea to reality. In the second part, the Appendixes, we can find some documents related to the project.

The first part is composed by eight chapters. Chapter 1 is a common chapter included in both volume I and volumen II. This chapter gives a complete problem analysis and how it was tackled. The analysis is formalized through a study of the requirements in Chapter 2. In Chapter 3 is specified the software architecture that will support the system; in Chapter 4 is made the design of the program, both internally, database and scheduling tasks, and externally, the interface. The Chapther 5 is to familiarize with manners, processes and tools adopted during Uburyo development, and Chapter 6 is to show the test proccess that verifies that the system meets the requirements listed in Chapter 2. Conclusions about all the work and future lines to continue are showed in Chapter 7. Finally, Chapter 8 includes all the references that support this work.

The second, and the last one, part is a collection of different documents. Chapter 9 shows the previous scholarship system form, that was one of the first studied document. Chapter 10 meets some presentations (slides and poster) that have been used to present the project at various events.

Before continuing with the report, it is important to say that all the pictures and diagrams used in the report (except figures 3.1 and 3.2 and different logos) are own designs.

CHAPTER 2

REQUIREMENTS ANALYSIS

2.1. Software Requirements Specification

2.1.1. Introduction.

This section is a Software Requirements Specification (SRS), for the Sustainable System of Loans. Its contents has been elaborated with TEDECO's cooperation group, performing the client role and transferring us necessities and experiences from the University of Ngozi. This specification has been structured following the advices written inside the standard "IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998" (Section 2.1.1.3).

2.1.1.1. *Purpose.*

The goal of this section is to define in a clear and precise way all the minimum functionality, as well as constrains, of the Sustainable System of Loans. Thus, this section was the communication tool among all the implied parts. Contents inside this section were elaborated among every member interested in the system.

This software specification has been reviewed by our client and by the development team as many times as it has been needed. These re-viewings have generated some different SRS versions until getting agreement by all the implied parts. This section explains readers that commitment.

2.1.1.2. *System Environment.*

One of the main problems in the developing universities is that registration fees, that students must pay in order to study, are too expensive compared with how much money one Burundi inhabitant may earn during one whole year. Therefore, many students are not able to pay these fees and they must stop their studies. Another problem is corruption in institutions which move money in these kind of countries.

Some years ago, the University of Ngozi offered students scholarships for increasing the number of people who come to Ngozi for studying in this university. In the other hand, the university did not have enough money for doing this so students should give back these scholarships if they wanted to get their diplomas. It was not a good attempt and many of these scholarships was not paid. Therefore, one of the main issues from which this project starts is that generally: Scholarships models do not work in developing universities because not enough budget; and Loans models, in which students must give back, the loaned quantity, just in cash are not sustainable too, because saving culture is not something related with this kind of countries. Here is where this project attempts to give another solution more specific to these cultures.

In TEDECO appears the idea of creating a Mini-Loans (Table 2.1) system for helping students to pay their registration fees in the local universities of their countries, as well as offer them another alternative way to give back the loans besides giving the money back. This is a Sustainable System of Loans because it attempts to find a solution for solving the problem of giving back these debts. The main idea is to look for an alternative way of returning the loans that is advantageous for students and for the university itself. This alternative way is returning the loan working. These jobs generate indirect or direct

TABLE 2.1. SRS Definitions

Term	Definition
Loan or Mini-Loan	Small quantity of money assigned to any university student in order to help him with the registration fees. But always with the commitment that the student will give back that money performing predefined jobs for the university. During the whole document will be used in the same way both Loan and Mini-Loan.
Sustainable System of Loans	Process applied in the university in order to help students to pay its registration fees. This process attempts to find an alternative solution for solving the student debts. The main idea is to offer to students another way to give back the money. This alternative way is to perform different task for the university.
Mini-Loan Application	Set of data necessary for applying. All these data is described in Application Form (Section 2.1.1.3).
Evaluation Commission	Set of people dedicated to evaluate student applications and to accept or reject new proposed jobs. Due to commission tasks nature, is interesting that this commission will be close enough to the project in order to be efficient with its task, and far enough for not being affected with its decisions. Thus, it will be interesting to create a heterogeneous commission, including academic institution staff and also external members, in order to achieve more transparency.
Job	Service needed by the university performed for any student in order to solve his debt. Jobs will be classified into three different groups: Jobs that increase the university value, Jobs that save money for the university and Jobs that generate incomes for the university (For deeper information see Section Economical Implications in Uburyo volume II [RAM10]).
Administratives	People who works at the university administration and participate in the administrative part of the mini-loans process and jobs administration.
Evaluation	Two weights assigned to each application attending two criteria: Economical and Academical. Each weight includes a comment related its justification.
Announcement Budget	Quantity of money available for the mini-loans in a concrete announcement.
Sustainable Loans Account	Book-keeper account for registering assigned money to the sustainable loan process.
Loan Quantity	Quantity of money loaned to any student.
Mini-Loans process resolution	List of students with assigned loan and money loaned to each one in any announcement.

incomes for the university. TEDECO has thought three different kinds of jobs:

- Jobs that increase the value of the university like computer maintainers, laboratory helpers, auxiliary teachers, etc.
- Jobs that save money to the university like working in the library, administration, etc.
- Jobs that generate incomes to the university like courses the university may offer to external students (People from companies, public administrations, etc.), students working in companies that pay to the university, etc.

The Sustainable System of Loans manages all this process in an efficient way and as much automatic and transparent as it is possible. It offers a web site for entering all the needed data for creating Mini-Loan applications. It also creates a common space in which the evaluation commission gives its opinion and evaluates each application and job. It generates also an employment bureau where students (with loans) may check jobs and their characteristics. And finally, it allows to follow the status of each student loan as well as manage all the information necessary for developing this process. In summary what it attempts is to integrate an employment bureau with a mini-loan manager. It does not try to communicate all the agents with the system. This means that if a teacher wants to include a new job in the employment bureau, he will do it via one administrative undertaken of entering and validating data in the system. It is not (by now) a collaborative tool neither in which commission members are able to communicate themselves by many ways. The software just gets commission member evaluations and it makes them accessible for every commission member.

With this tool we attempt to manage every needed data for performing the loan process, as well as generating transparency during the process, allowing to audit every action executed in the system and controlling the money that comes in and out in the sustainable system account.

2.1.1.3. *References.*

- IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE std. 830, 1998
- Application Form for the University of Ngozi, Burundi (Chapter 9).

2.1.1.4. *SRS General Vision.*

This section is composed of three subsections (2.1.1, 2.1.2 and 2.1.3). Subsection 2.1.1 is the introduction and it gives to readers a general vision of what is the utility of the SRS. The second subsection (2.1.2) contains a general system description in order to know the main functions that the system must perform, all the data the system is managing, factors, restrictions, suppositions and dependencies that affect to the system design but without going further. Subsection 2.1.3 are defined in a precise way all the requirements the system must satisfy.

2.1.2. General Description.

In this subsection is presented a high level description of the Sustainable System of Loans. It will be presented the main problem to solve, and functions that the system must perform. How the system will support these problems and the information needed for doing that. It is also described the found restrictions that affect to the design and another factors that influence in the system development.

2.1.2.1. *Product Perspective.*

Most of the software generated by TEDECO, in its project TICAMEN, is being installed right now. Software generated inside this project is for university management.

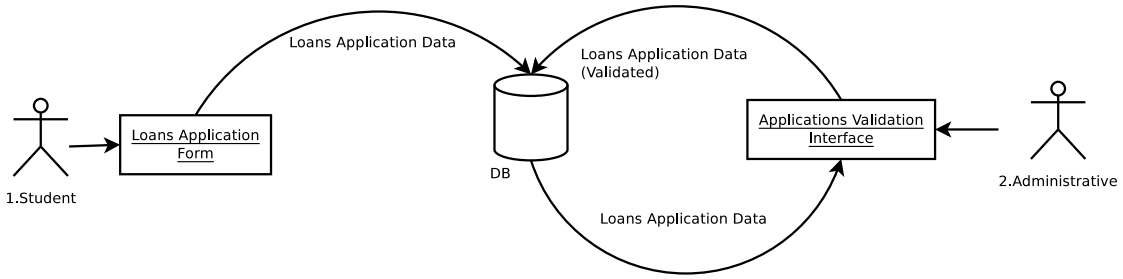


FIGURE 2.1. Loan Application Management

Due to many of the university managing processes are related and also they share data, it could be interesting to think about connecting all that independent applications for getting a more powerful university management and a very important point: transparency in processes.

For example, if this year are installed all these application in the UNG and its staff begins to make use of them, it could be a chance to look for students that work the next year in the inter-connection of them.

Readers may notice a direct relationship between the loan assignation (Section 2.1.2.4) and quantities giving back (Section 2.1.2.7) processes, from the Sustainable System of Loans, with university registration fees management, from the university accountancy system. Inter-connection between these two modules would give to the Sustainable System of Loans a direct and transparent repercussion into the university accounts.

2.1.2.2. *Product Functions.*

In general terms, the Sustainable System of Loans, must provide support for the following task in order to perform the loans system management:

- Loan Application Management
- Loan Assignation Management
- Loan Quantities Assignation Management
- Employment Bureau Management
- Loan Quantities Giving Back Management
- Loan Monitoring Management
- Auditing and Security

2.1.2.3. *Loan Application Management.*

Loans application has always implied providing many personal data. And not only providing data, but the validation and authentication of them by administration members is a compulsory task to get a fair assignation process. Thus, we may notice that loans application process is not a trivial task but a difficult collecting process, validation and authentication of the information.

Loans application process is mainly as the following. Students introduce needed data in the system, among them the ordered quantity. Students could do this by themselves or helped by an administrative. It is only necessary a computer with access to the system. Who is introducing the data is not important, the system simply will have an input data interface. This implies that it is not necessary any kind of log in in order to access this interface. Otherwise, with all the student data, the system gets a log in and password that, if the student is chosen like a borrower, they allow him to access to his personal interface (Monitoring loan interface) where he could find information about his loan status.

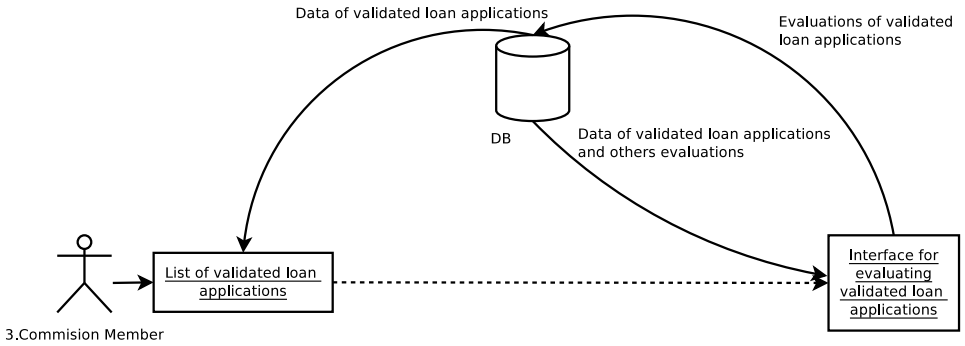


FIGURE 2.2. Loan Assignation Management

Once the student introduces his data in the system, he will pass to the validation and authentication period. Obviously, only electronic data is not enough for applying and it is necessary to demonstrate that this data is true showing different documentation that verify its reality. Students will present all this information in the university administration and, once everything will be verified, administratives will validate this electronic data. Administratives could modify any student data only if it is extremely necessary. When application are validated they are considered candidates to get the loan and, therefore they will be evaluated by commission members.

Obviously every university administration worker will get a log in and password for accessing the system and performing all these tasks.

The system will allow two configurations: Automatic, in which system administrators will configure the system to close the application collecting period in some concrete date (It could be enlarged); and manual, in which system administrators will close the application collecting period when they click some button or similar mechanism. Once collecting period is over the system will delete automatically all the non-validated applications.

System administrator will have the possibility of delete any application even they were validated.

2.1.2.4. *Loan Assignation Management.*

Once the loans application period is over, the system stores a set of applications validated by the university administration. For the loan assignation, commission members must have the possibility to access to all the data contained in each application. Thus, they could evaluate or classify in some way those applications, writing why this evaluation as well as some additional information that they consider necessary. All these evaluations must be visible for all the commission members.

For accessing to the list of validated applications, every commission member must be registered in the system. Needed data to register a commission member is: Name, electronic mail addresses, telephone numbers, qualifications and current job.

When commission members access to this interface, he will find a list of applications waiting for their evaluation. To perform the evaluation is to generate two ordered lists of applications, following two criteria: Academic and Economic criteria. As well as these two ordered lists, each commission member could introduce attached comments to each application, explaining or justifying the reason for locating each application in each position of the list. The ordered lists will not have to contain all the applications. It is possible that some commission member does not be able to evaluate some application because he does not have knowledge about the student. The system will allow to include in the ordered list just the application commission members want to evaluate. Therefore,

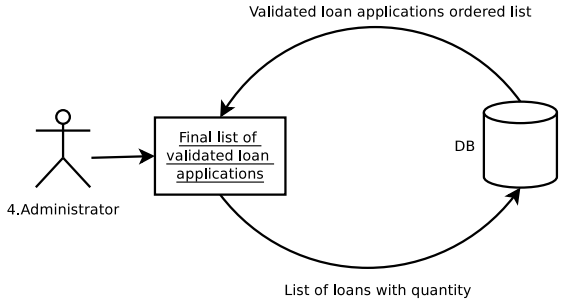


FIGURE 2.3. Loan's Quantities Assignation Management

the ordered lists will contain a number of elements equal or smaller than the number of validated applications.

Due to it is not necessary the evaluation of all the applications by the commission members, final position for one particular student, obviously will be calculated depending of haw many evaluations his application has received.

The system will allow two configurations for closing the evaluation period: Automatically in a pre-defined date or manually activated by the system administrator. If the system is configure for closing automatically the evaluation period, it will give the possibility for enlarging that period.

Once the final loan application list is generated, this will be accessible also to administratives, because during all the evaluation period, just commission members and system administrators have rights to check and modify these data. In this point begins the quantities assignation process.

2.1.2.5. *Loan Quantities Assignation Management.*

When the final list of applications is ready inside the system then it begins a new process period. This is the quantities assignation period. We already get an ordered list in which the first application has the highest priority for getting the money and the last one has the lowest priority. And we have an announcement budget too (Table 2.1). Depending of asked money by the students from the list head and the announcement budget, the system will assign more or less loans. This means that the system will assign the asked quantities from the beginning of the list until the announcement budget will be over.

It is necessary to emphasize that the system will be totally flexible in this part. Although by default it assigns the asked quantities to the first 'n' applications (Until the announcement budget is over), it must allow to change these quantities, even it will allow to assign money to any application out of the first 'n' applications. Therefore, I could reduce the quantity of the application 'n' in order to assign this reduction to the application 'n+1'. Once the loan's quantities assignation process is discussed by the commission and all the quantities have been set by the administrator according to the commission opinion, it will be generated the final application list with quantities, called Mini-Loans process resolution (Table 2.1). The system will store with this resolution also a report with all the validated applications that did not get money (Including the position where the applications were located in the resolution).

2.1.2.6. *Employment Bureau Management.*

As it was explained before, the loans process may be considered sustainable because the idea of giving back the money, not in cash but performing some tasks which allow the university to get profits. Once students get the loan they must give back the loaned money working for the university. Above it was commented that the entity which pro-

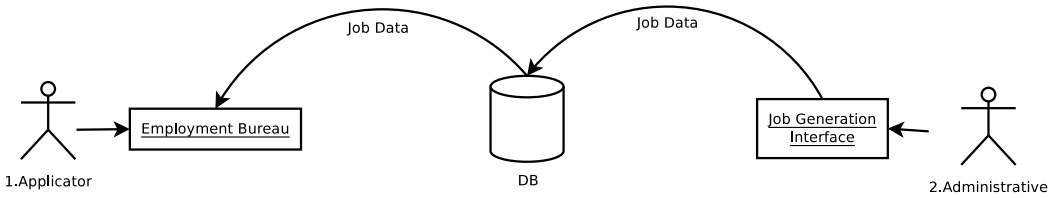


FIGURE 2.4. Employment Bureau Management

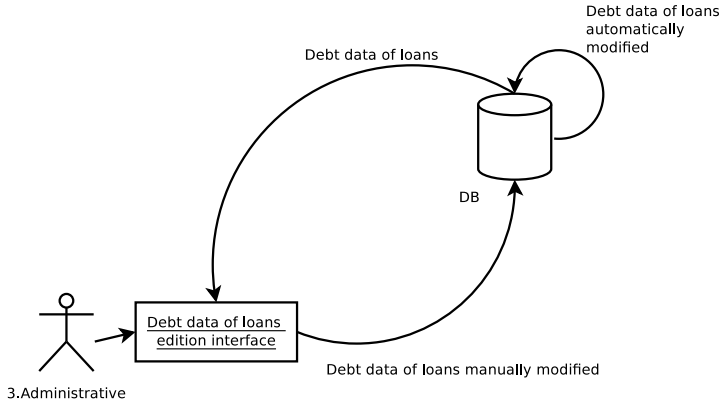


FIGURE 2.5. Loan's Quantities Giving Back Management

poses the job is diverse enough, from a teacher or librarian who needs an assistant til a university external company that needs employers. For this reason every job proposal will be communicated to the university administration and it will register this job in the system through a data input form.

Once a job is inserted in the system, every commission member will be warned and they will have a supervision period during they will be able to reject it for any reason they consider. This period may be configurable in the system. Every time a commission member rejects a job he must include a comment explaining why he rejected it. If new jobs exceed the supervision period without rejection they will be part of the employment bureau.

Henceforth, these jobs will be visible for all the applicators in the employment bureau. Consequently, an applicator could choose any of the jobs adapted to his requirements, contacting directly with the job promoter entity. Obviously, contact data of that entity will be available in the employment bureau. When the promoter entity, after evaluating all the interested applicators, takes a decision it will notify it to the university administration in order to update the employment bureau.

Rejected jobs by any commission member during the supervision period will be notify to administratives who communicate to the promoter entity the reasons of this rejection. Like this, promoter entities will use this feedback in order to propose any other job.

The proposals generation process is constant, this is that university administration is receiving new jobs at any time. A job inside the employment bureau could not be modified in any way.

Finally, and once the applicator finish the job, the promoter entity will sign a document assuring the applicator gets the goals. This document will be got by the university administrator and it will update the applicator status.

2.1.2.7. *Loan Quantities Giving Back Management.*

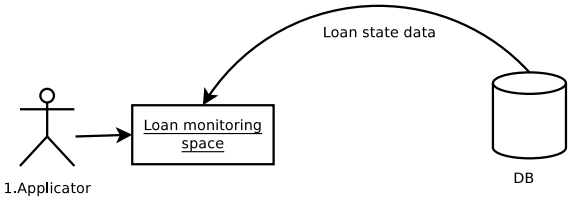


FIGURE 2.6. Loan Monitoring Management

Due to loaners have several ways to give back their assigned quantities to their applications, the system must dispose a generic mechanism that allows to management quantities giving back in the more homogeneous way.

Before, it has been commented that, right now, exist three different kind of jobs which generate even direct or indirect profits to the university:

(1) Jobs that increase the university value, like teacher assistants, computer maintainers, laboratory assistants, etc. This kind of tasks, for which the university is not hiring people (Apart from the students of the system), do not generate any direct profit but they give to the university more services and they make it more attractive for the students. The system must enter this profit as the quantity the job is reducing to the applicators debt.
(2) Jobs that save money for the university, like librarians, secretaries, painters, plumbers, etc. This kind of jobs do not generate any direct profit but they save money to the university. Instead of hiring external people for performing some task the university try to find a loaner with skills enough for executing that work. Although for these tasks the university do not get any external income, it must enter which quantity it is saving because it does not hire an external worker. This quantity does not have to be the same as the applicator debt reduction.
(3) Jobs that generate incomes for the university, like teachers of courses (with profits motives) for the society, employers in external companies, etc. In these cases the university is getting money for the job performed by applicators. In the first example the university gets money by the registration fees and in the second one it is getting all or part of the applicator salary. The university must enter this direct profit and, like in the second point, it does not have to be the same as the applicator debt reduction.

The quantities giving back ways, as readers may notice, are diverse but summarizing there would are: Constant and monthly amount or amount for any concrete task. Thus, the loans system will offer administratives two quantities giving back configurations: Automatic and manual. This configuration will be established by administratives in the job assignation form. Automatic configuration allows to assign to any job an amount that will be reduced to the applicator debt monthly or quarterly. Manual configuration exempts the system of reducing applicator debts, leaving this task in administratives hands who reduce applicator debts as much as they like and when they like through a loans system interface. Although, a job will be configure as automatic quantity giving back, administratives could reduce the debt manually or even cancel it. This configuration is allowed because loaners could pay with cash their debts at any moment.

2.1.2.8. *Loan Monitoring Management.*

In the moment a loan is created begins its monitoring. With monitoring we are referring some space in which different roles may check the current loan status. Obviously, this space will be accessible just for the applicator which the data showed belongs and not for

others. But moreover it will be accessible even by commission members, administratives and system administrators.

Logically this space will be read only and in any case it will be allowed to edit data in this interface. And with current loan status we like to say, for example, applicator current debt, performed jobs, opened jobs, etc.

2.1.2.9. *Auditing and Security.*

Due to the Sustainable System of Loans is attempting to create a fair and transparent loans assignation process, it will be necessary to implement a security mechanism in order to assure which event was executed by which user. This mechanism will be implemented as an auditory service of the system performed actions.

System administrators will access to a set of possible system executed actions in which they will define a subset to be audited. This subset will be different depending on system administrator criteria. It will be defined moreover the auditing persistence. This is the time during an event will be registered in the system. Once these parameters are configured, the system will be able to register and storing those events considered important and performed by all the system users. Thus, in case of irregularity in the process, we will have the needed information for detecting the origin of this irregularity. Therefore, the system must contain an interface where it will show the result of this auditing and it will be accessible just for the system administrator role.

Due to the loans system philosophy is executing the assignation process in a fair and transparent way, there is a set of events considered delicate or important that will be supervised by more than one system role. But, both debt reduction or canceling and job ending or canceling are events put in hands of only one actor (Administratives), therefore the system will audit always these actions.

2.1.2.10. *User features.*

Due to system users, except administrators, will not have to know about computers, it must offer an easy and intuitive interface guiding users through its screens and offering them as much help as it is possible.

The Sustainable System of Loans has four different kind of users: Applicators, Commission Members, University Administratives and System Administrators. Each role will have different rights in the system and it plays on different requirements. Specific competences of each role could be read in Security Attributes (Section 2.1.3.8).

2.1.2.11. *Suppositions and Dependencies.*

Since the application will work using a free data base system with an HTTP client-server architecture, it will be necessary to install and configure a server where these services are available. Moreover, it will be indispensable that the entity where the system will be installed, will have the computers in which users will access to the application connected to the server.

Because the philosophy of the evaluation commission is to divide geographically up its members for getting objectivity in the loans assignation process, the entity where the system will be installed must have external connection via Internet in order to get commission members accessing from outside.

2.1.3. Specific Requirements.

In this section are presented, without ambiguity, the functional requirements that must be satisfied by the system. Every requirement explained here will be necessary and fundamental, therefore the system will not be valid if it does not get some of the requirements

here described. These requirements have been written following, among others, the important testability criteria. This is that in each requirement will be easily provable if the requirement is satisfied by the system.

2.1.3.1. *Functional Requirements.*

(1) Loan Application Management

Req(1) Each time a student wants to apply for a loan, while the application period, he will introduce his data in the system through a form (Accessible without log in) with the following information, see section 9.

Req(2) University administratives could check all the student applications with its status (Validated or not).

Req(3) University administratives must be able to modify the data of non-validated applications.

Req(4) University administratives will be able to validate applications.

Req(5) System administrators could cancel applications even if they are validated or not.

Req(6) The system will not allow to enter more applications once the application period is over.

Req(7) When the validation period is over (Even automatically or manually), the system deletes non-validated applications.

(2) Loan Assignation Management

Req(8) Once the validation period is over, every commission member is warned, via email, about the availability of the validated application list.

Req(9) All the validated applications could be visualized by commission members but not for university administratives during the evaluation period.

Req(10) Commission Members could sort the validated application list twice and following two criteria: Economic and Academic criteria. These orders will go from the highest priority to the lowest. It will not be necessary to include all the application in one order.

Req(11) Commission Members will have the option of including comments reasoning their evaluations.

Req(12) Every Commission Member could read evaluation comments from another Commission Members.

Req(13) The system will not allow to evaluate any validated applications once the evaluation period is over. This period is configurable and enlargeable by system administrators or even closed manually by their selves.

Req(14) The system will generate a final ordered list from all the created orders did by commission members, once the evaluation period is closed even manually or automatically in a predefined date.

Req(15) Since the ordered final list is generated it will be also accessible to university administratives.

(3) Loan's Quantities Assignation Management

Req(16) System administrators should configure the following data for each new loan announcement: Announcement Budget (Available money for the loans in this announcement).

Req(17) The system, automatically, will assign the quantities to the applications in the final ordered list head til the announcement budget is over. Like this we will get 'n' loans.

Req(18) The system will notify to commission members the default loans final list is ready via email.

Req(19) The system will contain a messages table where commission members could discuss about the default loans final list.

Req(20) System administrators could modify assigned quantities by default (following commission member opinion described in the messages table), even assign money to any application out of the first 'n', if and only if the addition of all the quantities is not higher than the announcement budget.

Req(21) Once quantities are configured, system administrators could generate the final loaners list with the loan quantities called Mini-Loans process resolution.

Req(22) The system will store, with the Mini-Loans process resolution, a report with the students with validated applications but without loan (Including the position they get inside the loans final list).

(4) Employment Bureau Management

Req(23) University administratives could upload form(s) for proposing jobs.

Req(24) The system will allow to download and print the form(s) for proposing jobs without log in.

Req(25) University administratives could introduce, at any time, jobs in the system entering the following data: Job title, esteemed job hours, job explanation, students requirements, economic assessment for students, produced income profit for the university, university saves du to job performance, tutor responsible data, job promoter information.

Req(26) Jobs introduced in the system during a period predefined by system administrators (Day, week, month, etc.), will be notified via email to commission members with a supervising deadline, also predefined by system administrators.

Req(27) Commission Members could reject proposed jobs, indicating the rejection reason and before the supervising deadline specified in the noticing email.

Req(28) Once supervising deadline is got, the system will notify to university administratives and also to job promoters (If its email address is defined inside the proposal) those rejected jobs and it will delete them from the system.

Req(29) The system will include, after the supervising period, inside the employment bureau those jobs without rejection.

Req(30) A job never could be modified.

Req(31) Applicators could visualize all the jobs available without log in. They could see all their data except the produce income profit for the university.

Req(32) System administrators could delete any job.

Req(59) (This requirement was included after finishing the elicitation process and this is because its code is not following the last one). System administrators could change the status of any job {Rejected, Approved,

Non Approved}.

(5) Loan's Quantities Giving Back Management

Req(33) University administratives could assign jobs to applicators. During the assignation they will indicate which debt (A student may apply in more than one announcement therefore he would have more than one debt) the job is reducing and quantity giving back configuration {Manually, Monthly, Quarterly}. Once a job is assigned it will not be shown in the employment bureau.

Req(34) University administratives could upload form(s) for indicating the end of a job.

Req(35) The system will allow to download and print the form(s) for indicating the end of a job without log in.

Req(36) University administratives could upload job finishing memory index(es).

Req(37) The system will allow to download and print the job finishing memory index(es) without log in.

Req(38) The system will offer two types of giving back configurations: Automatic and Manual. Automatic configuration allow to assign an amount to be discounted automatically to the applicators debt monthly or quarterly. Manual configuration exempts the system to reduce applicators debt and will be university administratives who will reduce these debts through a system interface, as much as they like and when they like.

Req(39) University administratives will indicate when some assigned job is finished.

Req(40) University administratives will reduce or cancel the applicators debt.

Req(41) University administratives will cancel some assigned jobs in case of applicators leave them.

(6) Loan Monitoring Management

Req(42) Each loaner will have a personal space where will be shown some details about his loan, his debt and the jobs he has performed or he is performing in that moment. This personal space, as well as loaners, will be also accessible by university administratives, commission members and system administrators.

(7) Auditing and Security

Req(43) System administrators could select among a list of actions that users may perform in the system those they would like to control or audit.

Req(44) The system will audit always and, without the possibility of canceling this auditing by any system role, the following events: Job end, Job canceling, Debt reduction and Debt canceling.

Req(45) System administrators will define during how much time (auditing life time) a performed and registered action will be stored in the system.

Req(46) The system will register each auditable event performed. This registration will be available during the auditing life time defined by system administrators.

Req(47) The system will offer an interface, only accessible by administrators, where will be listed all the auditable events performed with an age smaller that the auditing life time.

(8) System Users Management

Req(48) Students will have to select a log in and password in application form for, just in case of being chosen as a loaner, accessing to their personal space with this log in and password.

Req(49) University administratives must be registered in the system with a log in and password for performing all their tasks. Their registration will be done by system administrators.

Req(50) Commission Members must be registered in the system with a log in and password for performing all their tasks. For registering a commission member system administrators should indicate: Name, email addresses, telephone numbers, studies, background and current job. Their registration sill be done by system administrators.

Req(51) Both university administratives and commission members registration must be controlled by system administrators having them the competence of allowing the access or not.

(9) System Administration and Parametrization

Req(52) The system will allow to its administrators the possibility of changing the language interface, between English and French, as well as implementing and adding new languages not included by default in the system.

Req(53) System administrators could configure the loans application deadline in two different ways: Automatic, in a concrete date (enlargeable); or Manual, executed by system administrators.

Req(54) System administrators could configure the validation deadline in two different ways: Automatic, in a concrete date (enlargeable); or Manual, executed by system administrators.

Req(55) The system will use an heuristic or formula editable by system administrators for generating the final ordered list, from all the particular orders made by commission members. The system will have an heuristic by default.

Req(56) System administrators could configure the period of time (Day, week, month, etc.) in which new jobs will be regularly notified to commission members.

Req(57) System administrators will define the job supervision period (By commission members) before being included in the employment bureau.

Req(58) System administrators could configure the evaluation deadline in two different ways: Automatic, in a concrete date (enlargeable); or Manual, executed by system administrators.

2.1.3.2. *User Interfaces.*

Uburyo users interface will be a web interface. It will have to communicate very different users among them and with the system. Most of these users do not use to work with computers and software applications. Thus, it must make easy the access to the desired information, the knowledge of which features it is offering, the execution of practical and specific tasks depending on the user role and, finally the navigation through different pages that are composing the web site used by the system to communicate itself with the users.

2.1.3.3. *Software Interfaces.*

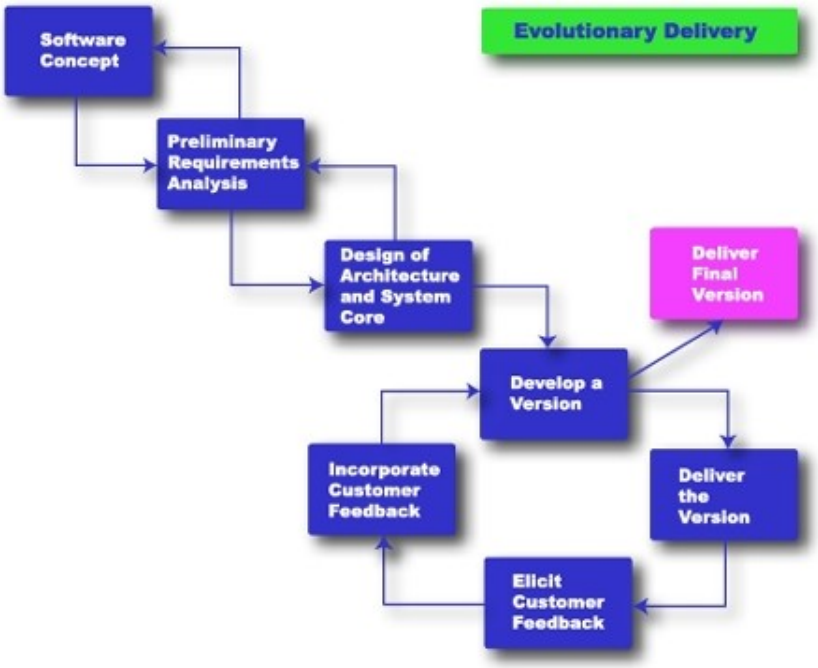


FIGURE 2.7. Evolutionary Prototyping

In its first version, this software will not communicate with any other software. This is that it will be completely autonomous. Although, TEDECO expects that, in a close future (Section 2.1.2.1), every product generated in different TICAMEN project stages works together for achieving a more powerful university administration. Thus, Uburyo design must be thought for the future and easy inclusion of API functions which communicate the system with outside.

2.1.3.4. *Communication Interfaces.*

Connection to the net will be established directly using the University local areal net. The system must have Internet connection for allowing the access to foreign commission members. Internet connection must be transparent to the system but due to poor electric systems from countries where the software will be installed, we will need to take care about frequent electric cuts which may get down the local areal net. Thus, the system must be prepared for net cuts.

Moreover, it is necessary to think that connection speed in this countries will be very slow, so the system will have to spend as less band width as is possible.

2.1.3.5. *Performance Requirements.*

The system will allow to be used by more than one user. It should exploit as many available technologies as it can, allowing the access as many users as the web server and data base support, and also keeping an average response time inside of the regular response time limits for this kind of systems.

2.1.3.6. *Development Requirements.*

The chosen life cycle for developing the product will be the evolutionary prototyping [WIK10e] in order to get growing constantly and make easier the inclusion of new changes and features.

2.1.3.7. *Technological Requirement.*

The system must be installed, in its version, in one of the servers of the University of Ngozi. In this case, the system should work on a server with the following characteristics:

- Processor: AMD Athlon(tm) 64 X2 Dual Core Processor 4800+
- Memory: 2 GB, RAM memory
- Ethernet net card

The operative system of this server is Debian Lenny Linux distribution. Therefore, the system must work on Linux.

This server has already installed Apache 2.2 HTTP server (apache2 meta-package) and server-side HTML-embedded scripting language PHP 5.2 (php5 meta-package) . Thus, the application interface should work using this technologies or others similar.

Regarding to data base, this server has already installed MySQL 5.1 data base server (mysql-server meta-package). Then, the application must use this data base manager in case of needing data base.

2.1.3.8. *Security Attributes.*

As has been commented before in User Features Section (Section 2.1.2.10), the Sustainable System of Loans count with different user types, among them: Students, university administratives, commission members and system administrators. And not all of them are registered in the system in same manner. Next, it is detailed how each kind of user is registered inside the system.

(1) Students: When students fill up the application form, they also introduce a log in and password. Like this, in case of being selected like a loaner, they will keep registered inside the system with that data. Else, their data will disappear from the system data structures.
(2) University administratives: They will be registered using a log in and password by system administrators.
(3) Commission members: They will be registered inside the system with data described in Req(48). Their registration will have to be validated in some way by system administrators.
(4) System administrators: First system administrator will be registered during the system installation. The followings system administrators will be registered by other system administrators.

Obviously, each user role will access to its own resources and functionality. Next, it is explained competencies of each role.

(1) Students: They could register in the system only one application but if they are loaners they could register more, one in each announcement. Moreover, they could access to their personal space if they are loaners too and they could access to the employment bureau both if they are loaners and not.
(2) University administratives: They could validate applications as well as edit those non-validated applications. Moreover, they could could access both ordered applications final list and mini-loan process resolution. They will have the possibility of accessing to the employment bureau and the competence of both introducing jobs and finishing or canceling them. They could also edit all loaners debt. Finally, they will have access to all loaners personal space.
(3) Commission members: They could visualize all the validated applications as well as generate two orders with all of these applications. Obviously, they could access to the final order as well as the mini-loans process resolution. They will also have access to the employment bureau and the competence of rejecting income jobs. Finally, they could access to loaners personal spaces.

(4) System administrators: They could delete applications both validated and not. They will have the competence of configuring the system with a collecting and validation deadline always enlargeable. They will have access to the non-validated and validated lists. They will have to configure a deadline for application evaluation always enlargeable. They will have access to the final applications order and they have the competence of assigning quantities to the loans as well as configuring the announcement budget. Obviously, the will have access to the mini-loans process resolution and the employment bureau as well as loaners personal spaces. They will also register university administratives, commission members and system administrators. Finally, they could perform all the assigned competences to university administratives.

2.2. Elicitation process

"In requirements engineering, requirements elicitation is the practice of obtaining the requirements of a system from users, customers and other stakeholders. The practice is also sometimes referred to as requirements gathering.

The term elicitation is used in books and research to raise the fact that good requirements cannot just be collected from the customer, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial because you can never be sure you get all requirements from the user and customer by just asking them what the system should do. Requirements elicitation practices include interviews, questionnaires, user observation, workshops, brain storming, use cases, role playing and prototyping." [WIK10g].

Uburyo requirements elicitation process has been basically based on interviews. In this process just stakeholders participated: Developers and TEDECO which could be consider as our customer but not our users. Uburyo users are mainly located, in this first scholarship process ,in Burundi and that made communication with them a difficult task. Interviewing TEDECO's members, and using their previous experiences in user analysis and software developing for under-developed countries, we achieved an almost complete software requirements specification. From the beginning of this software development stage, we were aware that getting a complete software requirement specification was an impossible challenge, without final user feedback and due to innovative process facet. Therefore, the following philosophy was adopted: We wanted to take an evolutionary prototype (Figure 2.7) and design and develop an easily modifiable code.

Another good practice adopted was to create a requirements management matrix [RM10a] once a final software requirements specification was achieved. Thus, it was easy to follow individually each requirement and know the status of Uburyo development at any time. In table 5.1 readers may see Uburyo development status at the date of this document. This table is automatically generated by the requirements management matrix depending on each requirement status.

Readers may notice that development status is not totally complete and it is 88,14% complete. This means that not all the requirements have been completely implemented, even that there are some requirements that definitely have not been implemented. This is because feedback got from first Uburyo deployment at University of Ngozi in Burundi. Next, we will summarize the reasons why seven requirements are not implemented. Requirements will be referenced by its code defined in a Software Requirements Specification (Section 2.1.3):

- Req(7): It is not clear if that is the best point during the process to delete automatically non-validated student applications. Automatic tasks are very sensible point in order to assure data consistency.
- Req(22): Reports have been considered as a second implementation phase when every main feature has been tested in real environments as University of Ngozi.
- Req(28): This requirement has been defined as a bug (Issue 32, see table Appeared Uburyo Bugs: Definitions from Uburyo volume II [RAM10]). It is specified in Uburyo bug tracker system [TED10a] and it will be assigned to local Uburyo support team (See Subsection Technical Agents from Uburyo volume II [RAM10]). This is explained in next chapters.
- Req(38): Manual configuration is already implemented but not automatic one. This is because it is not clear if automatic configuration is useful and even secure.
- Req(45) and Req(57): Scheduled tasks are not implemented due to local server date/time configuration. At the date of this document we cannot assure that this parameter will be well configure in local servers. Therefore, it is very insecure to schedule tasks without be sure that date and time will be well configured. However, scheduled tasks module are designed in section 4.3.
- Req(55): This requirement has been consider as a future feature because it is not necessary at this development point.

And next, we will explain the reasons why there are 12 incomplete requirements:

- Req(1): This is a style issue. The following points are pending to be solved by local Uburyo support team (See Subsection Technical Agents from Uburyo volume II [RAM10]) :
  - It is necessary that body content does not pass on the gray slide of the head.
  - It is necessary to center the titles.
  - It is necessary to include the Home bar on the body bottom.
- Req(14): The same reason as Req(55).
- Req(26), Req(29), Req(46), Req(53), Req(54), Req(56) and Req(58): The same reason as Req(45).
- Req(42): This requirement is planned to be completed by local Uburyo support team (See Subsection Technical Agents from Uburyo volume II [RAM10]).
- Req(44): This requirement has been defined as a bug (Issue 41, see table Appeared Uburyo Bugs: Definitions from Uburyo volume II [RAM10])
- Req(52): English and French languages are already implemented. At the date of this document is possible to include new languages but not from a software interfaces. How to include new languages are explained in the System Administrator Manual from Uburyo volume II [RAM10].

Although Uburyo development status is not 100% completed, software gets all the specified goals in SRS, it is totally usable and it supports completely Uburyo processes.

CHAPTER 3

SOFTWARE ARCHITECTURE

As many Web Applications over Internet, Uburyo is just a data base with an interface where some different kind of users generate events that extract or modify the information contained in it.

"Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [IEEE 1471]"[EEL06].

Thus, when you think about software architecture for Uburyo, you are thinking about software architecture for Web Applications.

Researching over Internet you may find a set of patterns facing many different problems.

"[An architectural style] defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style defines a vocabulary of components and connector types, and a set of constraints on how they can be combined." [EEL06].

And taking the UML definition "[A pattern is] a common solution to a common problem in a given context."[EEL06].

The most used pattern for Web Applications like Uburyo is the Model-View-Controller (MVC)[SUN02]. Model-View-Controller was described for the first time by Trygve Reenskaug [REE10] in 1979 when he was working in Smalltalk[WIL04] within Xerox PARC [PAR10].

"In the MVC paradigm the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task. The view manages the graphical and/or textual output to the portion of the bit-mapped display that is allocated to its application. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. Finally, the model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller)."[BUR92].

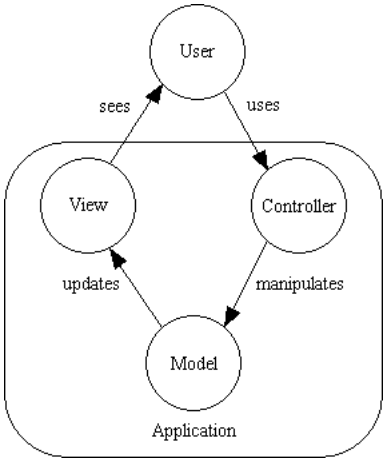


FIGURE 3.1. The basic MVC relationship [MAR04]

We can see an overview about relationships among different components in the Figure 3.1.

3.1. Model-View-Controller Pattern

Although the reader could find out many information about the Model-View-Controller Pattern on Internet, in this section we attempt to show a brief explanation of what is it interweaving that our research has achieved to join. This theory is the basis of Uburyo Pattern.

Following the UML definition we may reinforce that a Software Pattern is an efficient solution to a common problem in a given context. There is a computer science branch that studies Software Patterns following the philosophy: If you need to solve some problem, probably someone has solved it before. Therefore, describe a Software Pattern is describe a context, a problem, a solution and consequences of applying this solution.

Usually, the problem solution description is done by a diagram showing the pattern components, its responsibilities and relationships. And sometimes the problem solution description contains as well some strategies for applying this structure to our design. Thus, every pattern specification well-defined must be written following this structure.

3.1.1. Model-View-Controller Context.

Application is accessed by a defined set of different users and it presents different information in numerous pages depending on what kind of user is requesting the information. Moreover, the engineering team dedicated to design, implement and maintain the application is compose of different skills individuals.

3.1.2. Model-View-Controller Problem.

Nowadays, enterprise applications need to present the information in many different ways depending who is checking that information. For example, a managing store application must be accessed by Internet Users, in order to buy products; Sellers, for registering products they sell; Managers, for checking everything is correct; Administrators, for creating users, changing passwords, etc.

Usually, all these different users access to the same data container but depending on what kind of user is accessing, he will have different rights and information will be shown in a different way. Normally, these accessing rights are organized in accessing levels. Thus, an user with a higher accessing level, could performance the same actions

than an user with a lower accessing level, as well as more specific actions belonging to his accessing level.

When developing application has to face up against single user applications, sometimes is beneficial to interweave data access, business rules logic and user interface. However, this approach becomes inadequate when developed systems must support multiple roles. Breaking applications into three parts for managing data access, business rules logic and user interfaces, makes easier implementation, testing and maintenance as well as re-using components and software developing.

MVC paradigm is based in how users interact with applications: Input (Controller), Processing (Model) and Output (View).

3.1.3. Model-View-Controller Forces.

- Enterprise Data needs to be shown in different ways and using different technologies: HTML, XML, Swing...
- Enterprise Data needs to be updated via different interactions: Users link clicking, URI requesting using curl [CUR10], SOAP messages written in XML...
- Supporting multiples type of views, interactions and store options must not impact in components that provide application functionality.

3.1.4. Model-View-Controller Solution.

3.1.4.1. *Structure.*

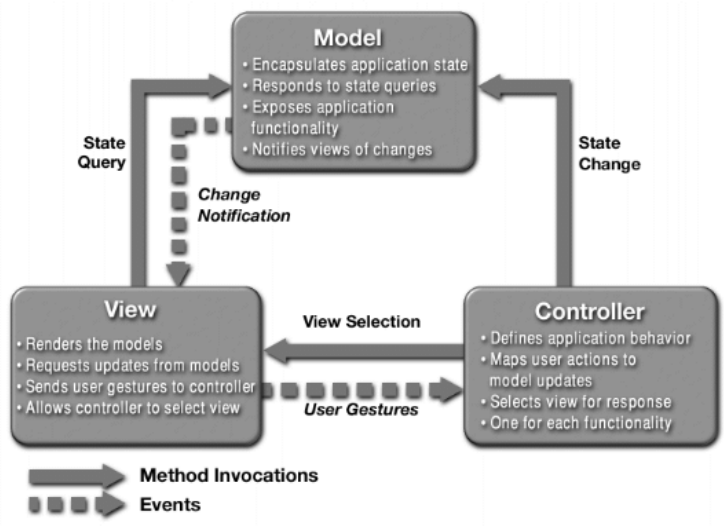


FIGURE 3.2. MVC Diagram Solution [SUN02]

The Figure 3.2 represents a blueprint for solving the problem described before.

3.1.4.2. *Participants & Responsibilities [MAR04].*

- Model
  - A model is an object representing data or even activity, e.g. a database table or even some plant-floor production-machine process.
  - The model manages the behavior and data of the application domain, responds to requests for information about its state and responds to instructions to change state.

  - The model represents enterprise data and the business rules that govern access to and updates of this data. Often the model serves as a software approximation to a real-world process, so simple real-world modeling techniques apply when defining the model.
  - The model is the piece that represents the state and low-level behavior of the component. It manages the state and conducts all transformations on that state. The model has no specific knowledge of either its controllers or its views. The system itself maintains links between model and views and notifies the views when the model changes state. The view is the piece that manages the visual display of the state represented by the model. A model can have more than one view.
- View
  - A view is some form of visualization of the state of the model.
  - The view manages the graphical and/or textual output to the portion of the bit-mapped display that is allocated to its application.
  - The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented.
  - The view is responsible for mapping graphics onto a device. A view typically has a one to one correspondence with a display surface and knows how to render to it. A view attaches to a model and renders its contents to the display surface.
- Controller
  - A controller offers facilities to change the state of the model. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate.
  - A controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response.
  - The controller translates interactions with the view into actions to be performed by the model. In a stand-alone Graphic User Interface (GUI) client, user interactions could be button clicks or menu selections, whereas in a Web application they appear as HTTP GET and POST requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.
  - The controller is the piece that manages user interaction with the model. It provides the mechanism by which changes are made to the state of the model.

3.1.5. Consequences.

- Re-use of components: Since the design is divided into three isolate components and data accessing process is doing always in the same way, it is easier to re-use components generated applying this pattern.
- Easier functionality developing:
  - Due to all business logic is concentrated in the Controller Component, it becomes easier to re-use, modify and include new features in applications.
  - Due to interface specification is allocated in the View Component, it becomes easier to modify it.

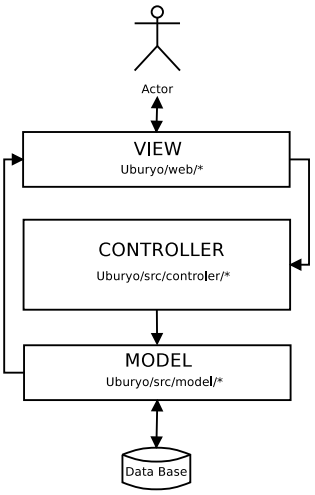


FIGURE 3.3. Uburyo Layer Diagram

  - Including new type of roles becomes simple. Just writing a new view and some controller logic and wire them into the existing enterprise application.
- Increased design complexity: MVC pattern force designers to organize the code in some extra classes due to the separation of model, view and controller.

3.2. Uburyo Pattern

Once MVC pattern is understood is clear to realize how Uburyo components are organized and how they interact. In this section it will be explained relevant components in order to understand communication between users and application. How Uburyo changes and shows its state.

Uburyo PHP files are mainly organized into three important folders:

- uburyo/web/
- uburyo/src/controler/
- uburyo/src/model/

Web and Controler folders include more over the same folder structure and they keep its files depending for which actor are dedicated them (Students, Applicators, Administratives, Commission Members and Administrators). Therefore, inside the web and controler folders exist the following sub-folders:

- student/*
- applicator/*
- administratives/*
- commission/*
- administrator/*

Figure 3.3 shows a brief description about how Uburyo component are organized following a layer pattern.

3.2.1. Uburyo Controller Layer.

Management Applications are composed mainly by two kind of interfaces: Data input and data output interfaces. Users change the model state modifying the input controls included in input interfaces. Otherwise, users check the model state requesting output interfaces. Following this behavior Uburyo controller manages input and output data using two different classes: Form and View. Each data input interface reports data to

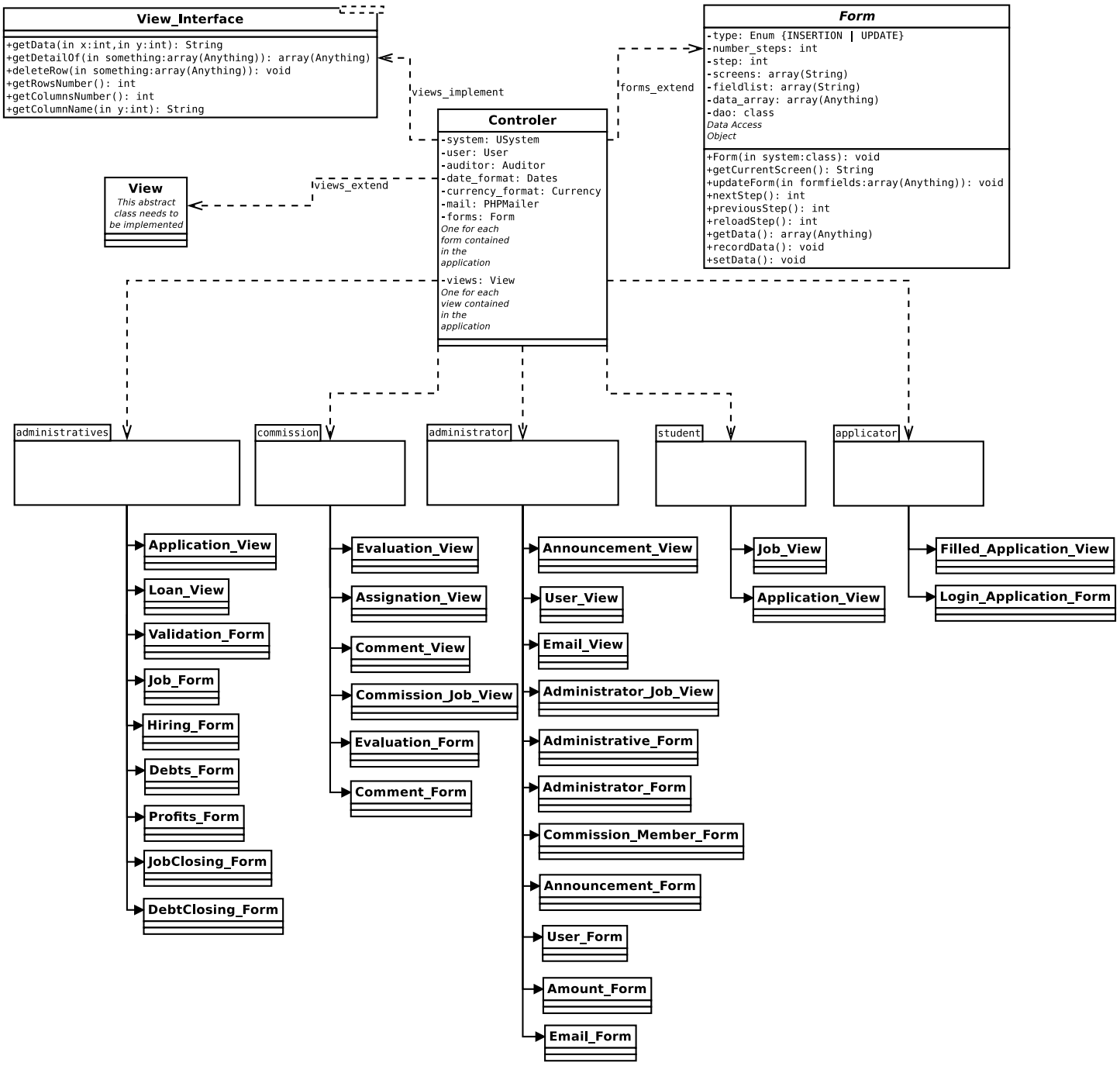


FIGURE 3.4. Uburyo Data input and output Management Diagram in Controller

a specific class inherited from Form. And each data output interface is nourished by a specific class inherited from View. Therefore, in Figure 3.4 Uburyo data input and output management is performed by, as many classes as different input and output user interfaces it has.

At the data input and output management diagram are represented every form and view class that compose the application at the release date of this document.

In addition, readers may notice the View Interface component. This component is implemented for supporting communication from the model to the view. Normally, views in Uburyo are lists of elements but in many cases users needs to get details from one concrete element. For this purpose is created the interface view. Model layer uses this interface in order to get the knowledge about how to manage view objects.

Otherwise, as is shown in Figure 3.5, controller layer contains also classes for performing other processes like log in, final application list generation, employment bureau management, email sender, etc.

3.2.2. Uburyo Model Layer.

Uburyo model implements communication between business logic and data. It represents data entities and relationships among them. Therefore, model components throw queries to the data base, get result-sets, process them and send data to controller components or directly to view components.

Model layer contains a super class called Default_Table. Communication between controller components and data in Uburyo follows always the same structures. Thus, the application builds queries following always the same rules. These rules are implemented in Default_Table class.

Therefore, Default_Table class specifies how to build select, insert, update and delete SQL statements and then model uses one specific class for each data base table. These specific classes are called Data Access Objects (DAO's) and they contain concrete knowledge for building queries in order to exploit each data base table. And all of these specific classes inherit query building knowledge from Default_Table. Readers may see these relationships in Figure 3.6.

In this way, when Uburyo Controller needs to get or change the model's state it creates as many DAO's as tables it has to exploit. Thus, every concrete form and view, as is shown in Figure 3.7, use different DAO's in order to communicate itself with the model state. Figure 3.7 also contains middle classes like Application, Evaluation, Loan and Commission_Member for unify treatment among different DAO's. I.e. when students apply for a loan they must fill out an Application Form, submitted data from this form is kept into three different data base tables: User, Student and Application. Middle classes unify this treatment in order to reuse it.

As well as form and view classes change data with model components, other classes that support Uburyo business logic also need to use DAO's. In Figure 3.8 is described this data exchange.

3.2.3. Uburyo View Layer.

This package as well as controller package is separated into five different sub-packages. Thus, interfaces are organized into different folders depending on which user will use that interface:

- student/* : for when the user is a student with a loan
- applicator/*: for when the user is a student who has made an application for a loan

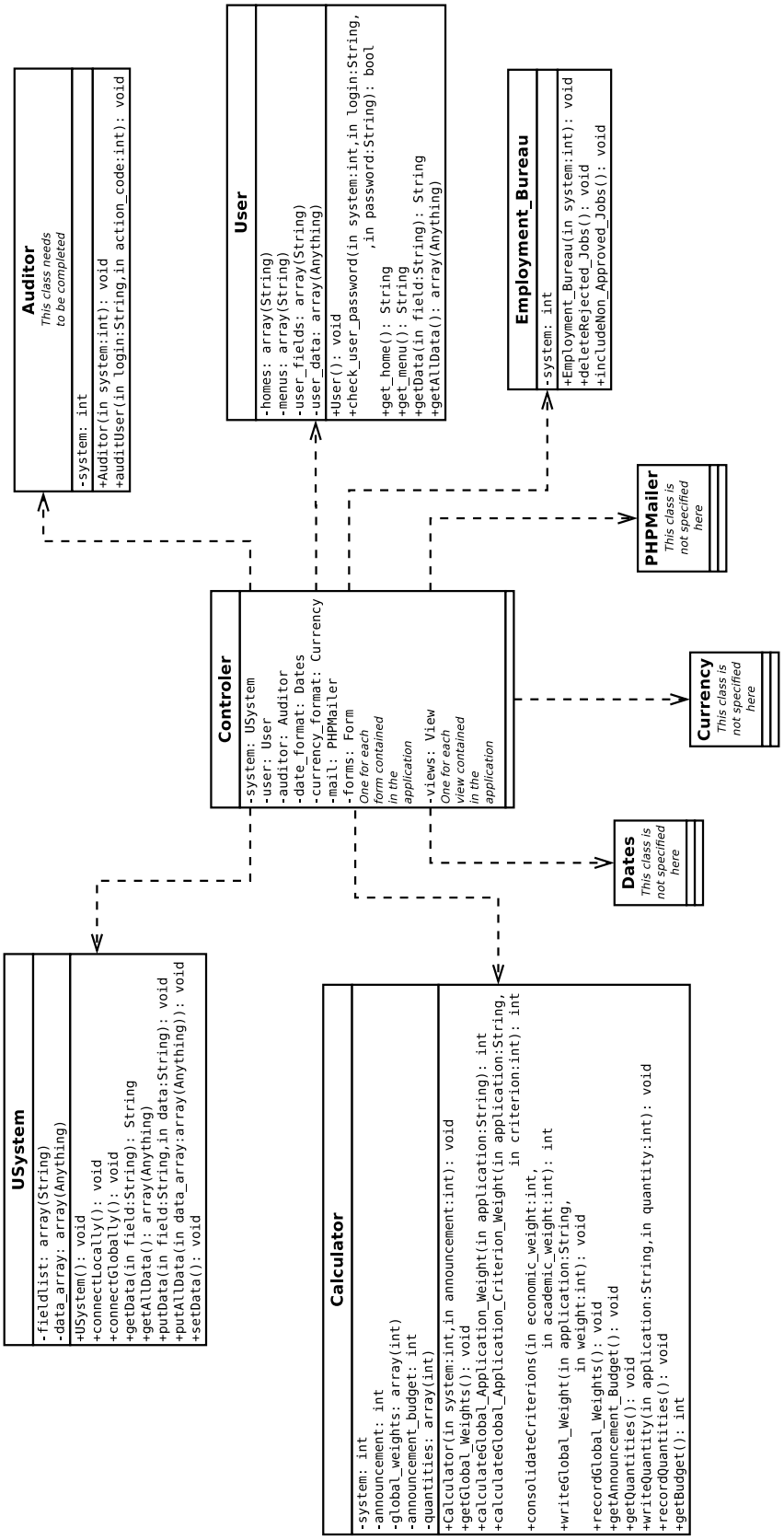


FIGURE 3.5. Controller supporting classes Diagram

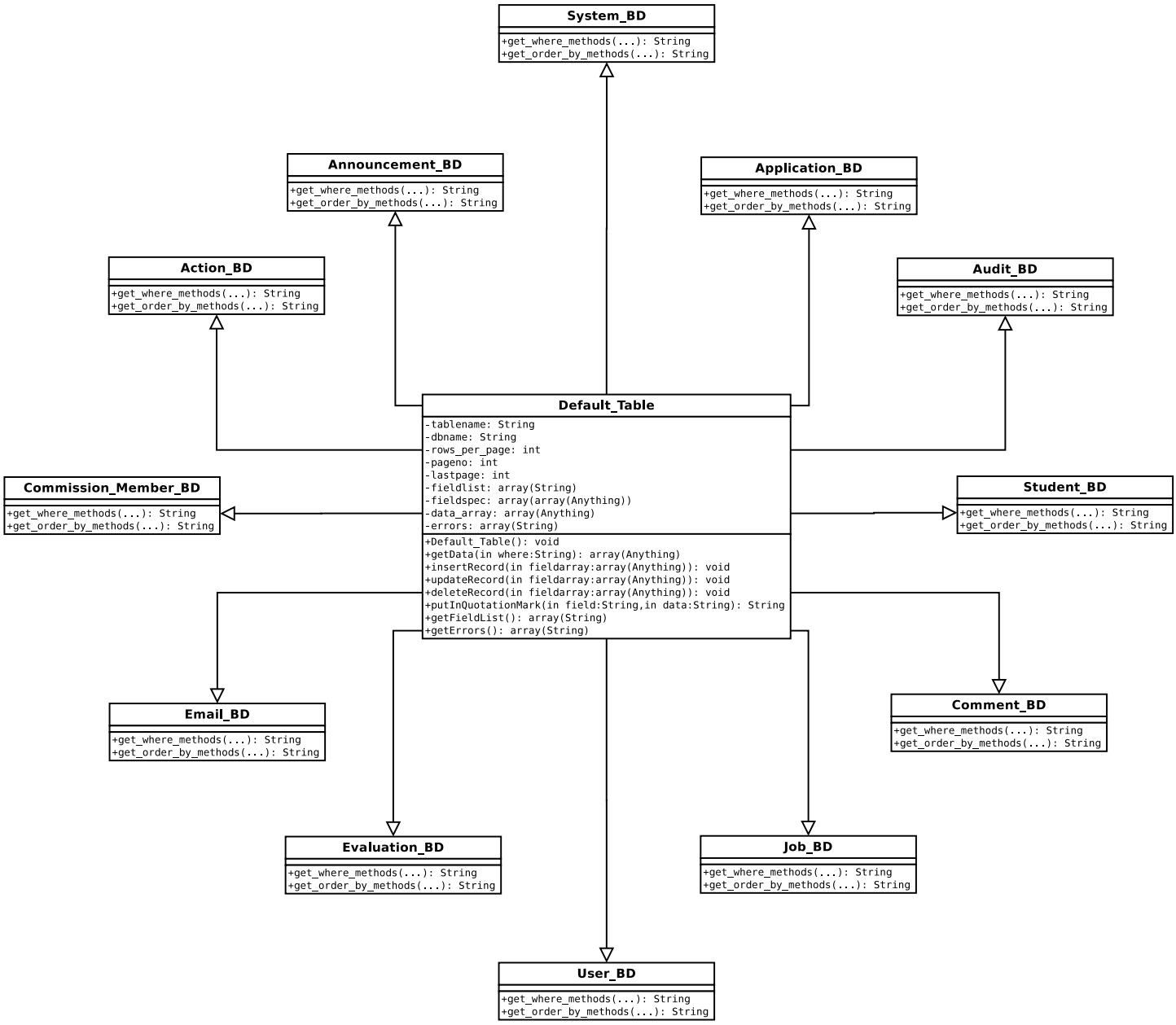


FIGURE 3.6. Model supporting classes Diagram

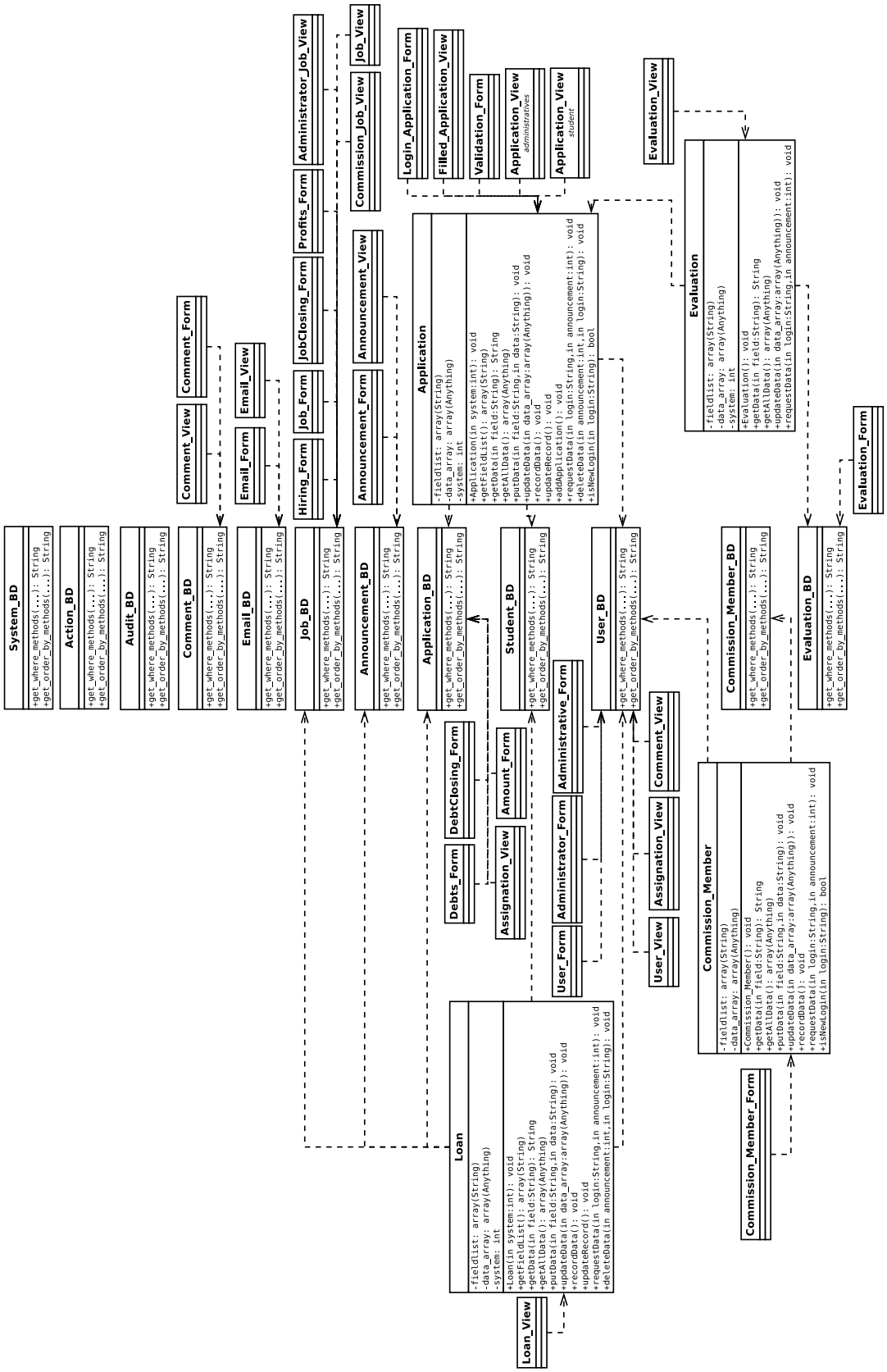


FIGURE 3.7. Model Data input and output Management Diagram

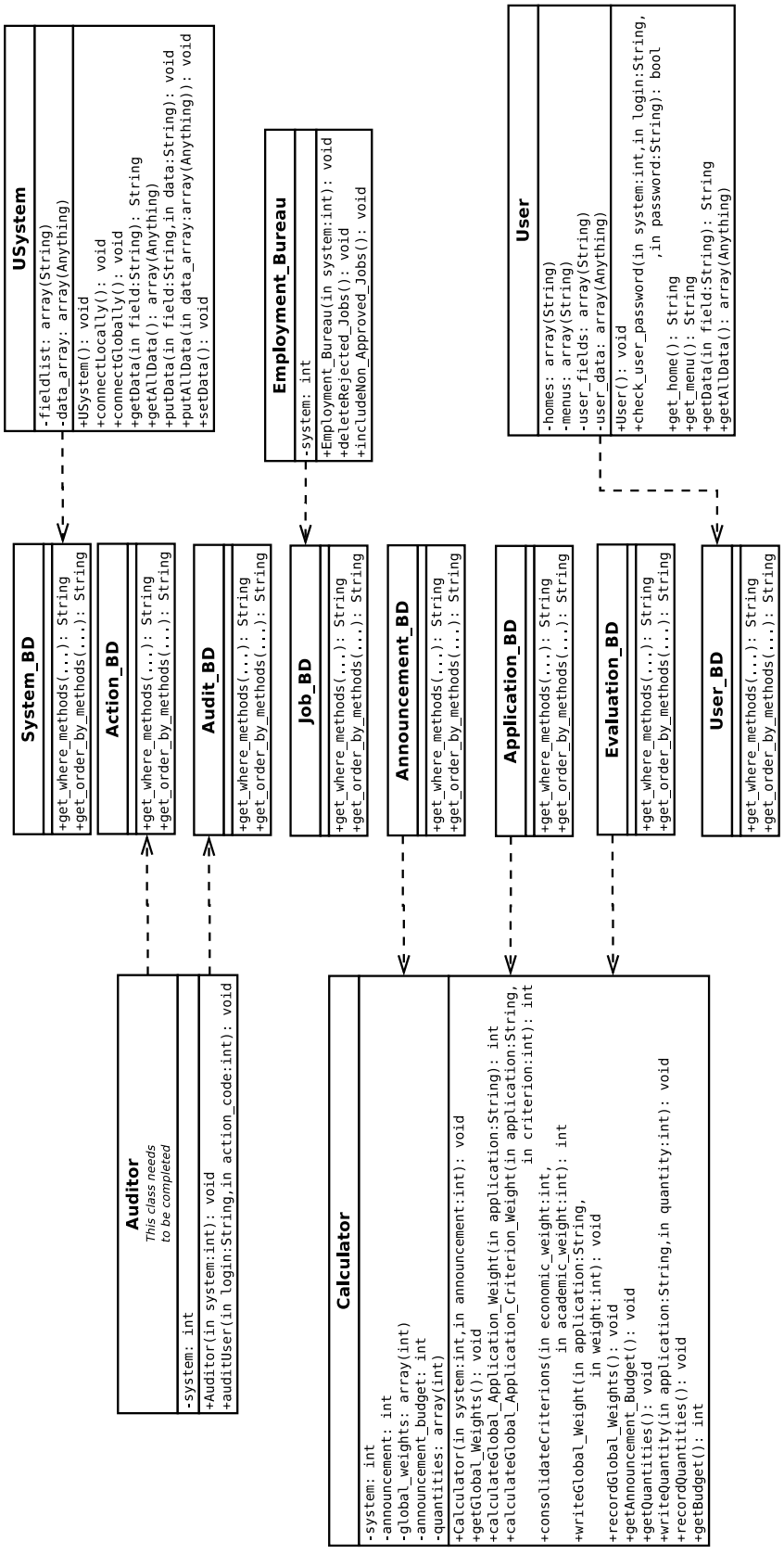


FIGURE 3.8. Controller supporting classes communication with DAO's

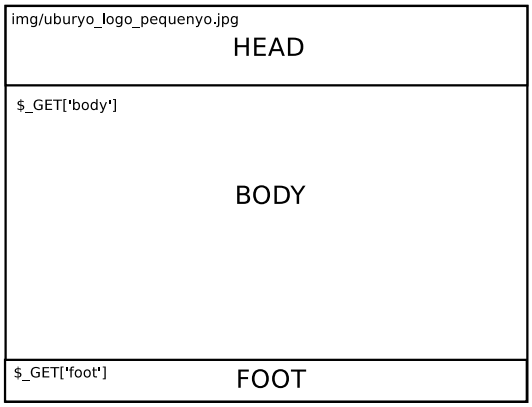


FIGURE 3.9. Frame for non logged Users

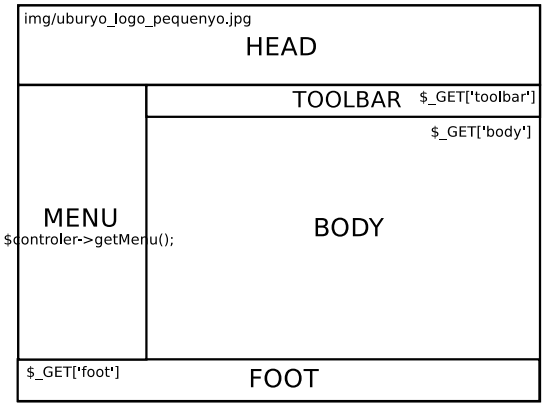


FIGURE 3.10. Frame for logged Users

- administratives/*: for when the user is an administrative of the educational center
- commission/*: for when the user is a member of the evaluation commission
- administrator/*: for when the user is an administrator of the system

All these interfaces are loading into two different frames. One frame for non logged users and another for logged users.

With Figure 3.9 readers are able to understand how HTML body is divided into three areas. Different information is shown inside these areas for non logged users.

And Figure 3.10 shows how HTML body is also divided for logged users but readers may notice this kind of users find more areas on the screen.

Every view sub-package is composed by one menu element and many body elements. Depending what kind of user is accessing to Uburyo he will be able to perform different actions. These actions will be executed by users using menu items. And these executions will load different body elements into the area reserved for this purpose.

Elements that belong to applicator, administratives, commission and administrator sub-packages are loaded using logged users frame. Although, elements inside student sub-package are loaded employing non logged users frame. Figure 3.11 describes how view elements are loaded into designed frames.

View Layer uses another relevant element that is important to explain here. This component is called the Announcement Selector Toolbar. Many of the information view layer shows inside the body area depends on the announcement. Therefore, many of the Uburyo views use the announcement filter and select the information belonging to the

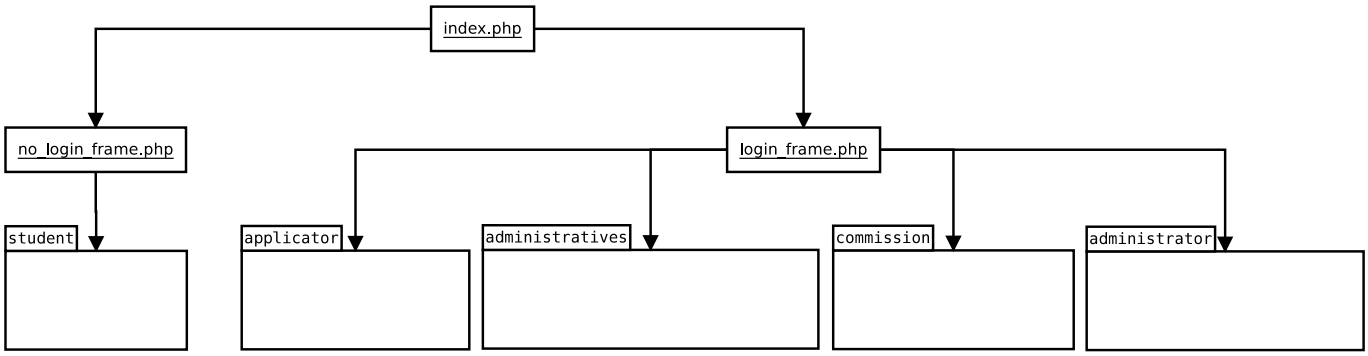


FIGURE 3.11. View Layer sub-packages

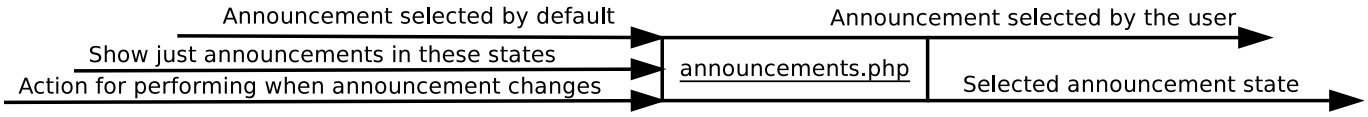


FIGURE 3.12. Announcement Selector input and output variables

selected announcement. Users select what is the announcement they like to check using the Announcement Selector Toolbar. This element is located into the toolbar area and it is a HTML select element. Figure 3.12 indicate which information Announcement Selector Toolbar needs for building up the selector and which information is returning by user events.

3.3. Interface-Database Communication

As it was explained in Uburyo model layer, communication between Uburyo and data base is done by DAO's. Uburyo implements one DAO for each data base table. And these DAO's inherit knowledge from Default_Table element. In this section is described how Default_Table connects with data base, execute queries (select, insert, update and delete) and manage SQL errors.

Uburyo is designed for executing simple SQL queries. This means that Uburyo generates queries just for exploiting one data base table. If it is necessary to exploit more than one data base table, Uburyo will generate as many queries as tables needs to exploit. Default_Table is a super class that implements the following methods:

- getData ($where);
- insertRecord ($fieldarray);
- updateRecord ($fieldarray);
- deleteRecord ($fieldarray);
- putInQuotationMark ($field, $data);
- getFieldList ();
- getErrors ();

Methods that generate queries are: getData for SQL selecting; insertRecord for SQL inserting; updateRecord for SQL updating and deleteRecord for SQL deleting.

3.3.1. Selecting Data with Uburyo.

```php
function System_BD () {
    global $kernel_bd;
    $this->tablename = 'system';
    $this->dbname = $kernel_bd;
    $this->rows_per_page   = 10;
    $this->fieldlist = array('system_code', 'language', 'job_collection_time', 'job_evaluation_time', 'date_format', 'currency',
        'automatic_evaluation', 'even_evaluation', 'email_sender');

    $this->fieldspec['system_code'] = array('pkey' => 'y', 'type' => 'INTEGER');
    $this->fieldspec['language'] = array('pkey' => 'n', 'type' => 'INTEGER');
    $this->fieldspec['job_collection_time'] = array('pkey' => 'n', 'type' => 'INTEGER');
    $this->fieldspec['job_evaluation_time'] = array('pkey' => 'n', 'type' => 'INTEGER');
    $this->fieldspec['date_format'] = array('pkey' => 'n', 'type' => 'STRING', 'size' => L_STRING);
    $this->fieldspec['currency'] = array('pkey' => 'n', 'type' => 'STRING', 'size' => L_STRING);
    $this->fieldspec['automatic_evaluation'] = array('pkey' => 'n', 'type' => 'INTEGER');
    $this->fieldspec['even_evaluation'] = array('pkey' => 'n', 'type' => 'INTEGER');
    $this->fieldspec['email_sender'] = array('pkey' => 'n', 'type' => 'INTEGER');
}
```

FIGURE 3.13. System DAO constructor

Uburyo generates SQL select statements always following the same structure: “SELECT * FROM $tablename $where_str $limit_str;”. For creating these queries Default_Table employs three variables: $tablename, $where_str and $limit_str.

Variable $tablename is in fact a class property and it is instantiated in constructors. Thus, each DAO implements its constructor assigning the data base table name it is exploiting.

After, variable $where_str is created concatenating the string “where” to the input parameter $where. $where is really a set of pairs column name and value, joined by logical operators like and, or, etc.

And finally, variable $limit_str contains the SQL limit query part. This variable employs the class properties $rows_per_page and $pageno and it is used for view pagination. This means that views in Uburyo show a pre-defined number of elements and this mechanism allows users to change among pages for see all the view elements. As well as $tablename $rows_per_page class property take value in constructors. Figure 3.13 shows a DAO constructor example.

Reader may realize at this point that Uburyo just build simple SQL select statements. And if any view is nourished by data from more than one table Uburyo will need to build as many select queries as tables needs to exploit.

3.3.2. Inserting Data with Uburyo.

SQL insert statement are built by Uburyo following this pattern: “INSERT INTO $tablename SET $fieldarray;”. In this case Default_Table uses the class property $tablename and the input parameter $fieldarray.

$fieldarray is a PHP array where indexes are the names of the table columns and values are data to be inserted (“column_name” => “column_data”). With this information Default_Table is able to build the insert statement.

3.3.3. Updating Data with Uburyo.

For updating information in the data base, Default_Table generates SQL update statements with the following structure: “UPDATE $tablename SET $update WHERE $where;”. For doing this, Default_Table makes use of two variables and the class property $tablename.

This method receives like an input parameter $fieldarray but, in this case, Default_Table creates with this PHP array two different variables: $update and $where. $update is a set of pairs column name and value and these will be the updated columns. $where is a set of

pairs columns name and value too although it defines the condition that rows should fit in order to be updated. For doing this separation, Default_Table employs the class property $fieldspec.

The property $fieldspec is an array of arrays in which is described each table column. For each column is created an array that usually contains the following information: If the column is primary key or not, column data type and column data size. Using the index pkey, Default_Table is able to know if the column analyzed in $fieldarray is primary key or not. If it is primary key this pair will be located in the variable $where but, if it is not, the pair will be located in $update. Figure 3.13 shows an example of how is define the property $fieldspec.

putInQuotationMark method uses the type index for embed data between apostrophes if data type is string.

3.3.4. Deleting Data with Uburyo.

For building delete SQL statements: "DELETE from $tablename WHERE $fieldarray;", Default_Table use the same philosophy than for inserting data.

3.3.5. Connecting to Data Base with Uburyo.

Each query SQL building method explained previously, before executing the built query against the data base, needs to connect with it. For doing this, Default_Table employs a function defined implemented in db_conf.php called db_connect($dbname). This function imports all needed information from the configuration file db_properties.php, in which is set up the connexion data like host, user, password, data base name, etc.

3.3.6. Handling SQL errors.

Uburyo defines an error handler and implements it in the file error_handler.php in the CONF folder. Function errorHandler($errno, $errstr, $errfile, $errline, $errcontext) takes SQL errors and give a brief description about the reason of it.

CHAPTER 4

DESIGN

4.1. Data Base

4.1.1. Entity-Relationship Diagram.

At Figure 4.1 readers may study which data entities compose Uburyo model and its relationships (At document's date). In this figure is only shown entities and relationships but not entity attributes. For checking attributes of each entity readers must continue reading Data Dictionary section.

4.1.2. Data Dictionary.

In this section will be described each entity attribute of every Uburyo entities (At document's date). These attributes will be separated in tables depending on which entity they belong.

Table 4.1 describes all the information Uburyo need for defining a user. A user could be an Administrative, Commission Member, Administrator, students that are applying and students that have got a loan in any announcement, called in Uburyo applicators.

Passwords are encrypted using md5 technology and the attribute type indicates which role performs the user in the system. For students, applicators and commission members Uburyo need also more information stored in student and commission_member tables (Tables 4.2 and 4.7).

For describing completely a student in Uburyo is necessary, as well as the information stored in user (Table 4.1), personal data for a better knowledge of them. Table 4.2 shows which indeed information Uburyo needs.

When a new loans process begins, administrators need to set up a new announcement. An announcement is composed by its description, different dates, budget and how will be generated the final application list. In table 4.3 readers realize what data is managing uburyo in order to support all the loans process by announcements.

Each student interested in getting a loan in any announcement should fill up an application form with different economic and academic data. This data will be employed by Commission Members in order to evaluate applications and it will be stored in application table, described in Table 4.4 and 4.5.

TABLE 4.1. User Entity Attributes

Attribute	Description	SQL type	Static	Key
login	System user identification	char(50)		PK
system_code	System parameters set code where user belongs	integer		PK FK
password	Every user needs the pair (login, password) to access the system	char(50)		
first_name	User first name	char(50)		
last_name	User last name	char(50)		

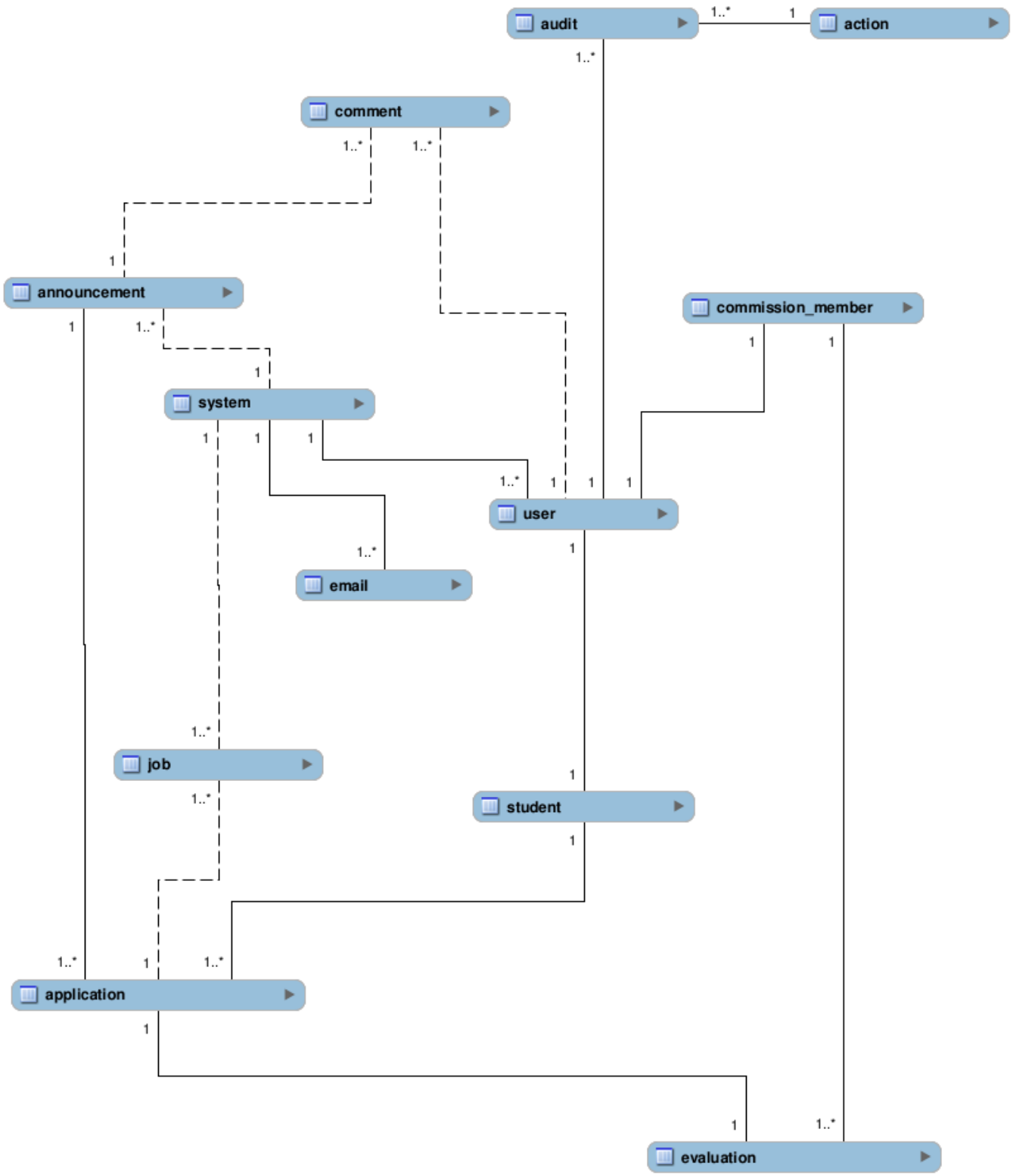


FIGURE 4.1. Summarize Entity-Relationship model at the document's date

TABLE 4.2. Student Entity Attributes

Attribute	Description	SQL type	Static	Key
login	System student user identification	integer		PK FK
system_code	System parameters set code where user belongs	integer		PK FK
number	University student number. Identification number within his university	char(50)		
birth_nationality	Country where the student is from	char(50)		
birthday	Student birthday	timestamp		
birth_province	Province where the student is from	char(50)		
birth_town	Town where the student is from	char(50)		
marital_status	Student marital status {Single, Married}	integer	Yes	
wife_name	If the student is married, this field will be his wife's name	char(50)		
nationality	Current nationality of the student	char(50)		
sex	Student sex {Male, Female}	integer	Yes	
residence	Location where the student is living nowadays	char(50)		

TABLE 4.3. Announcement Entity Attributes

Attribute	Description	SQL type	Static	Key
announcement_code	Announcement system identification	integer (autoinc.)		PK
system_code	System parameters set code where announcement belongs	integer		FK
date	Announcement date	timestamp		
description	Brief explanation of the announcement	char(500)		
budget	How much money has been assigned to the announcement	integer(big, unsigned)		
application_deadline	Date when the system will not allow to apply for a grant anymore, in that announcement	timestamp		
validation_deadline	Date when the system will not allow to validate applications anymore, in that announcement	timestamp		
evaluation_deadline	Date when the system will not allow to evaluate applications anymore, in that announcement	timestamp		
function	Global mathematical evaluation function that the system will use in order to get the final list	Char(250)		
state	State in which the announcement is {Collecting, Validating, Evaluating, Assigning, Ended}	integer	Yes	

TABLE 4.4. Application Entity Attributes

Attribute	Description	SQL type	Static	Key
login	System student user identification	char(50)		PK FK
system_code	System parameters set code where application belongs	integer		PK FK
announcement_code	Announcement system identification	integer		PK FK
date	Application starting date	timestamp		
state	Application state {Non_validated, Validated, Adjudged, Closed}	integer	Yes	
global_weight	Result of applying the mathematical evaluation function to the evaluations set of this application	decimal (10,2)		
quantity	If the application results chosen this field specify how much money has been assigned it	integer		
fathers_name	Student father's name	char(50)		
fathers_job	Student father's job	char(50)		
fathers_alive	Is the student father alive? {Yes, No}	integer	Yes	
mothers_name	Student mother's name	char(50)		
mothers_job	Student mother's job	char(50)		
mothers_alive	Is the student mother alive? {Yes, No}	integer	Yes	
fathers_address	Student father's address	char(250)		
humanities_diploma	Has the student got any humanities diploma?	integer	Yes	
state_diploma	Has the student got any state diploma?	integer	Yes	
state_diploma_organization	If the student has got a state diploma. Where has he got it?	char(50)		
state_diploma_section	What is the section within the organization where the student got the state diploma?	char(50)		
state_diploma_year	When did the student get the state diploma?	year		
state_diploma_results	What is the student state diploma's grade?	char(50)		
state_exam_results	What is the student state exam's grade?	char(50)		
ung	Has he studied at UNG?	integer	Yes	
ung_school	What is the school where he studied within UNG?	char(50)		
ung_department	What is the department where he studied within UNG?	char(50)		

TABLE 4.5. Application Entity Attributes (bis)

Attribute	Description	SQL type	Static	Key
ung_year	When did he study at UNG?	year		
ung_results	What was the student UNG grade?	char(50)		
other_university	Has he studied at other university?	integer	Yes	
other_university_name	If he has studied at other university, what was it?	char(50)		
certificate	Has the student got any certificate from the university where he studied in?	integer	Yes	
university_certificate	What is the university where the student got his certificate?	char(50)		
registration_year	Year in which the student is applying for a mini-loan	year		
registration_school	School where the student is registered at the moment he applies for a mini-loan	char(50)		
registration_department	Department where the student is registered at the moment he applies for a mini-loan	char(50)		
registration_option	Option taken by the student at the moment he applies for a mini-loan	char(50)		
registration_semester	Semester in which the student is applying for a mini-loan	char(50)		
registration_expenses	Money applied by the student to pay his registration expenses	integer (unsig.)		
maintenance_expenses	Money applied by the student to pay his maintenance expenses	integer (unsig.)		
other_expenses	Money applied by the student to pay other expenses	integer (unsig.)		
debt	If the application results chosen this field specify how much money the student must still pay	integer		

TABLE 4.6. Job Entity Attributes

Attribute	Description	SQL type	Static	Key
job_code	Job system identification	integer (autoinc.)		PK
login	Who is performing the job?	Char(50)		FK
system_code	In which system the job is set up?	integer		FK
announcement_code	What debt is reducing this job?	integer		FK
expiry	When will the job deleted in the system if any student has taken it	timestamp		
creation_date	When the job was set in the system	timestamp		
starting_date	When should the job be started by any student interested in	timestamp		
ending_date	When should the job finish by any student who takes it	timestamp		
value	Money that the job will reduce the student's debt	integer(unsigned)		
profits	Money that university gets due to the student's work	integer(unsigned)		
state	Job's state inside the system: NON_APPROVED, APPROVED, ASSIGNED, ENDED			
creator	Who is requesting a student for the job?	char(50)		
refund_type	How will be reduced the student debt, periodically by the system or manually by the administratives	integer	Yes	
hours	How many hours needs the job in order to be completed?	integer(unsigned)		
title	Job's title	char(50)		
requirements	What are the requirements that students need in order to get the job?	char(500)		
description	Job's description	char(500)		
reject_comment	If the job is rejected why it is	Char(500)		
earned	How much money has been earned yet performing the job	integer		

TABLE 4.7. Commission Member Entity Attributes

Attribute	Description	SQL type	Static	Key
login	System commission member user identification	char(50)		PK FK
system_code	System parameters set code where user belongs	integer		PK FK
background	Last commission member experiences, jobs or studies	char(500)		
organization	Organization what the commission member belongs	char(50)		
job	Where is the commission member working right now?	char(50)		
telephone_number	Commission member telephone number	char(50)		

TABLE 4.8. Evaluation Entity Attributes

Attribute	Description	SQL type	Static	Key
student_login	System student user identification	char(50)		PK FK
announcement_code	Announcement system identification	integer		PK FK
system_code	System where the evaluation has been done	integer		PK FK
commission_member_login	System commission member user identification	char(50)		PK FK
criterion	Each commission member evaluates each application following two criterion: Economical and Academical criterion	integer	Yes	PK
weight	Result of the commission member evaluation	decimal (10,2)		
comments	Reason because the commission member assign a result to an application	char(500)		

For introducing a new job in Uburyo administratives must fill up a form with different data in order to describe a job. Table 4.6 shows which information is needed for defining a job.

As it was explained previously some kind of users, like students, applicators and commission members, need some extra information for supporting the loans process. Table 4.7 describes which data Uburyo employs for managing Commission Members.

When validation step finish in any announcement, commission members must start to evaluate each application in order to get the final application list. All the evaluations done by commission members are stored in the table evaluation described in Table 4.8.

TABLE 4.9. System Entity Attributes

Attribute	Description	SQL type	Static	Key
system_code	System parameters set code	integer (autoinc.)		PK
language	System language configuration {French, English}	integer	Yes	
job_collection_time	The system will be collecting jobs during a time, for example 7 days. Every 7 days the system will send a notification to the commission members informing about new jobs. This time could be configured, and it will be saved here.	integer (unsig.)		
job_evaluation_time	How much time has the commission members to evaluate the student applications?	integer (unsig.)		
date_format	Date format used in the system. I.e. yyyy-mm-dd, dd-mm-yyyy, etc.	char(50)		
currency	Currency used in the system. I.e. $, FBU, etc.	char(50)		
automatic_evaluation	Is the automatic evaluation (combo box instead of input text) is set up {Yes, No}	integer	Yes	
even_evaluation	Is allow for commission members evaluate two or more application with the same weight {Yes, No}	integer	Yes	
mail_sender	Is the system sending emails when some actions are performed {Yes, No}	integer	Yes	

TABLE 4.10. Audit Entity Attributes

Attribute	Description	SQL type	Static	Key
date	When was the action performed?	timestamp		PK
login	What was the user who performed the action?	char(50)		PK FK
action_code	Performed system action code	integer		PK FK
description	-	-		

Uburyo is a software that needs some parameters to be configured and work in the good way. These parameters introduce flexibility and allow local administrators to adapt the loan process to their local institutions. Table 4.9 stores this parameters.

As Uburyo is a software that supports a loan process and it allows to perform this process with transparency, it needs to register each action performed in the system. Audit table, described in Table 4.10, stores all the action executed in Uburyo.

Every action that may be executed is stored in the action table. Therefore, this table is a static table. It will be loaded with data during the installation and Uburyo just execute select queries against this table. If developers include a new feature that needs to be

TABLE 4.11. Action Entity Attributes

Attribute	Description	SQL type	Static	Key
action_code	System action code	integer (autoinc.)		PK
description	Text action description	char(250)		

TABLE 4.12. Email Entity Attributes

Attribute	Description	SQL type	Static	Key
email_code	System email identification	integer		PK
system_code	System where email belongs	integer		PK FK
language	Language in which email is written {French, English}	integer	Yes	PK
subject	Email subject	char(250)		
body	Email body	char(500)		

audited they must insert a new row in this table. Table 4.11 describes this entity.

Uburyo could be configured for sending emails when users perform some pre-defined actions. These emails are stored in the table email. It is described in Table 4.12.

4.1.3. Managing Uburyo Data Development.

As all the applications, Uburyo is an alive software. Thus, it will be necessary to fix new appeared bugs, to develop new required features, to modify some interfaces and to create and delete new fields in views or forms. Therefore, from the beginning, Uburyo architecture was thought for an easy inclusion, modification and deletion of table fields and even tables itself.

Model-View-Controller pattern allows Uburyo to get this requirement. Any data base modification will be automatically detected by Uburyo if developers include this new information in the model layer. Consequently, inclusion, modification or deletion of any table field will involve the modification of the concrete DAO that represents the modified table. Once it is done, modification will be available either in controller layer or view layer.

Including new data base tables is something more complicated. Creation of new tables implies implementation of new DAO's for describing these new tables. But as field creation, once the new DAO is created, controller layer will be able to exploit this new data base table.

4.2. Interface

Uburyo software is structured basically following differences between users that must login into the platform and users that must not. Thus, readers may find two different groups: Students as non-login users and Applicators, Administratives, Commission Members and System Administrators as login users.

Students as non-login users have divided their interface into three different areas as is shown in figure 3.9: Head where a logo is loaded, body where are loaded their views and forms and foot where usually is located some navigator bar.

Otherwise, Applicators, Administratives, Commission Members and System Administrators as login users have their interface divided into four different areas (Figure 3.10): Head where in this case as well as a logo is included a welcome message and logging out

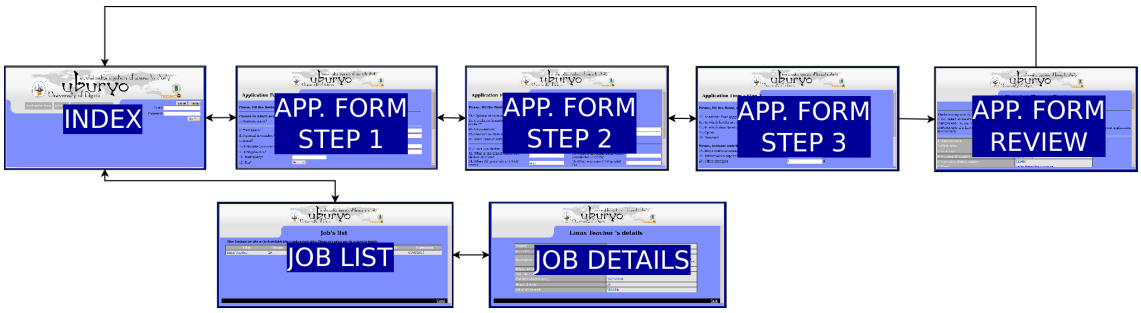


FIGURE 4.2. Student Web Map

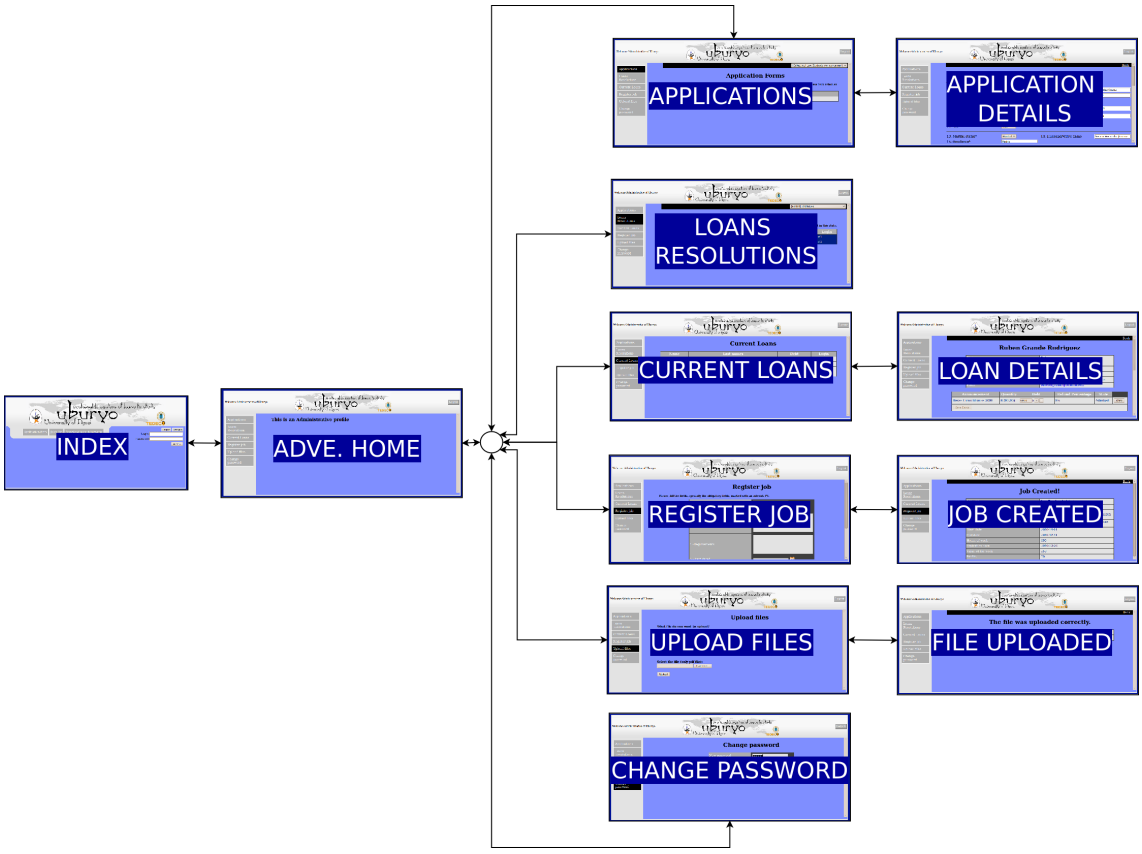


FIGURE 4.3. Administratives Web Map

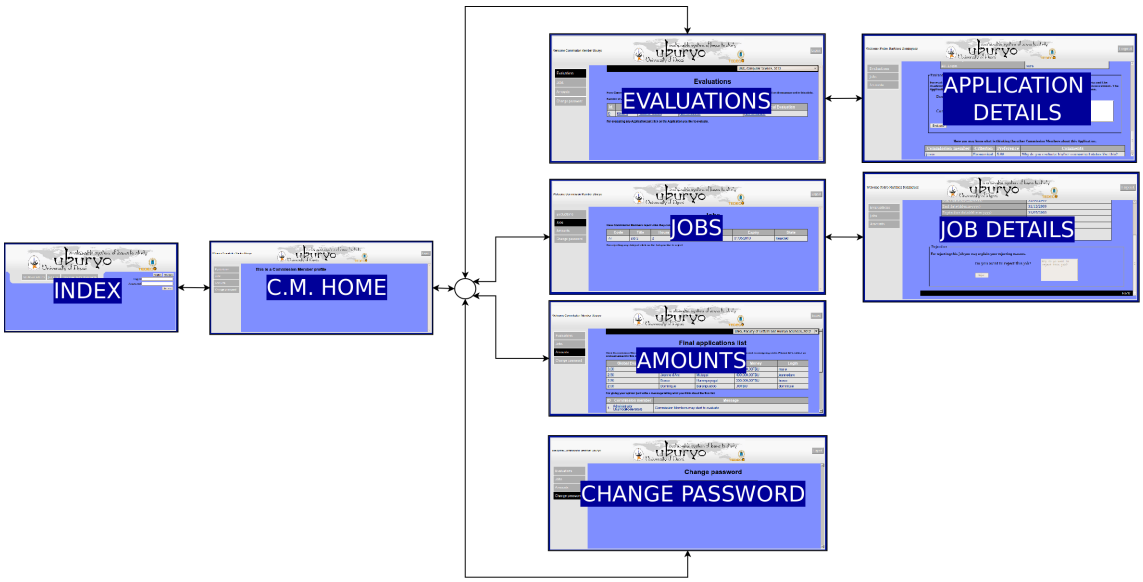


FIGURE 4.4. Commission Member Web Map

button, menu where different links are located to navigate by different views and forms that compose the user profile, body where the content is loaded and foot where is also located sometimes a navigator bar.

Moreover, all these interfaces must be translated to English and French (Section 2.1.3, Req 52). Uburyo interface multilingual is supported by Gettext from GNU..

"GNU 'gettext' is an important step for the GNU Translation Project, as it is an asset on which we may build many other steps. This package offers to programmers, translators, and even users, a well integrated set of tools and documentation. Specifically, the GNU 'gettext' utilities are a set of tools that provides a framework to help other GNU packages produce multi-lingual messages. These tools include a set of conventions about how programs should be written to support message catalogs, a directory and file naming organization for the message catalogs themselves, a run-time library supporting the retrieval of translated messages, and a few stand-alone programs to massage in various ways the sets of translatable strings, or already translated strings. A special GNU Emacs mode also helps interested parties in preparing these sets, or bringing them up to date." [GNU10b].

At the date of this document different user types interfaces are organized as following:

- Students Web Map shown in Figure 4.2.
- Administratives Web Map shown in Figure 4.3.
- Commission Member Web Map shown in Figure 4.4.
- System Administrator Web Map shown in Figure 4.5.
- Applicator Web Map shown in Figure 4.6.

4.2.1. Internationalization.

In computing, internationalization and localization mean to adapt computer software to different languages and regional differences. Internationalization is the process of designing a software application in order that it may be adapted to several languages and regions without engineering changes. Localization is the process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text.[WIK09a].

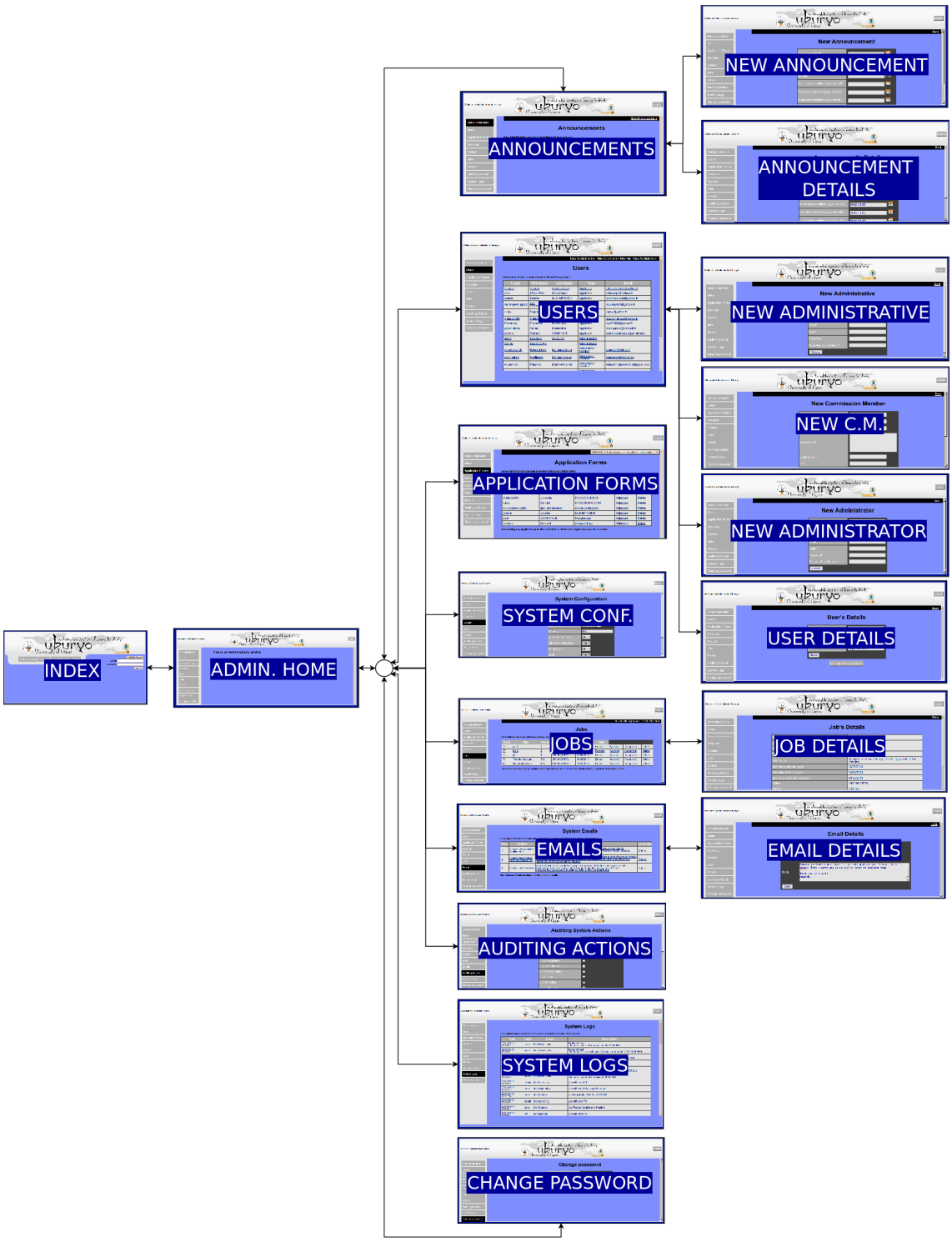


FIGURE 4.5. System Administrator Web Map

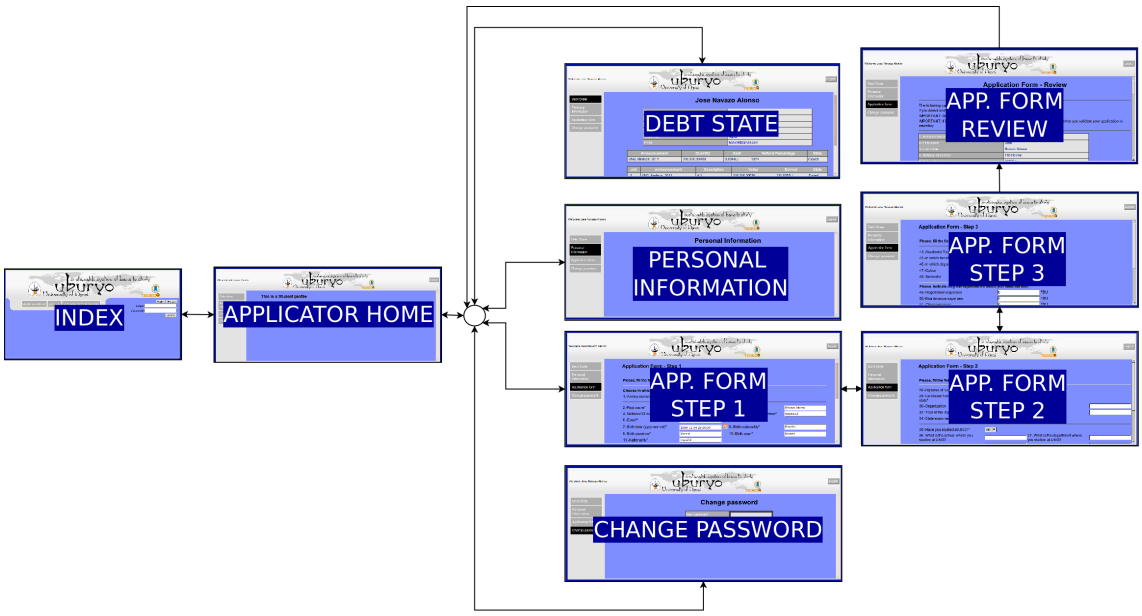


FIGURE 4.6. Applicator Web Map

In this project the internationalization is fundamental, mainly because is a cooperation project and free software. This means that this project will be use in many contexts around the world, being important that all people can use it easily.

For internationalization is used PHP-Gettext, a library that provides PHP functions to do a complete internationalization and location.[SAT07, PHP09].

The System Administrator Manual (in Uburyo volume II [RAM10]) has a section in which is explained how to use this library to maintain the system completely internationalized and at the day.

4.3. Scheduling tasks

As readers may see, in Uburyo software requirement specification there are some requirements which specify that some tasks must be executed automatically by the system. Some of them in a concrete date, others regularly. These tasks are the following:

- Closing automatically application period: Requirement 53.
- Closing automatically validation period: Requirement 54.
- Closing automatically evaluation period: Requirement 58.
- New jobs notifications: Requirement 56.
- Including automatically new jobs in employment bureau: Requirement 57.
- Deleting automatically old audits: Requirement 45.

At the date of this document and as readers may see in section 2.2 any job scheduler is still implemented in Uburyo. However, this Uburyo module is already thought and designed and this is the reason of this section.

As it was explained before, Uburyo version released, at the date of this document, is a web-based application independent to the operative system of the computer where it is installed. This is because this version does not make use of any service specific to the operative system. When we need to include a job scheduler mechanism in Uburyo software, we start to use this specific operative system services. In Linux, this service is called cron.

"Cron is a time-based job scheduler in Unix-like computer operating systems. The name cron comes from the word "chronos", Greek for 'time'. Cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. It is commonly used to automate system maintenance or administration, though its general-purpose nature means that it can be used for other purposes, such as connecting to the Internet and downloading email." [WIK10d].

But in Windows it is called task scheduler and it works in a different way.

"*Task Scheduler is a component of Microsoft Windows that provides the ability to schedule the launch of programs or scripts at pre-defined times or after specified time intervals. It was first introduced in the Windows 95 Plus! pack as System Agent but was renamed to Task Scheduler in Windows 98.*" [WIK10f].

Other operative systems will have its concrete task schedulers and they are not the purpose of our study. Here is explained a solution that does not restrict Uburyo software to a concrete operative system. However, its installation will be a little bit different in Windows than in Linux if job scheduler service is going to be used.

The main idea is to create a new class for checking one by one every different scheduled task commented above. This class will have mainly two public methods: constructor and startScheduler. The second one will call, as readers may see in figure 4.7, to other class methods for checking if any announcement gets any deadline, if system must notify new jobs, if some job has passed its supervision period and if is stored any old log.

Moreover, we will need a script for creating this class and also call to the method for starting the scheduler. This script will be called uburyo_crontab.php and it will be located in a new folder called API inside src folder. This will be the script executed by cron or task scheduler every night.

Next, we will explain how to program the execution of the script uburyo_crontab.php using Linux and Cron. This must be done during Uburyo installation if we need to use scheduling task. Therefor, after finishing the installation (Installation Manual in Uburyo volume II [RAM10]) let's execute in a terminal:

```
uburyo@server:crontab -e
```

A crontab file will be opened for edition. At the end of this file we will have to include the following line:

```
30 0 * * * php [public folder]/uburyo/src/API/uburyo_crontab.php
```

Like this, the server will execute everyday at 0:30 the PHP script uburyo_crontab.php which will create an instance of Job_Scheduler class and will check every scheduled task.

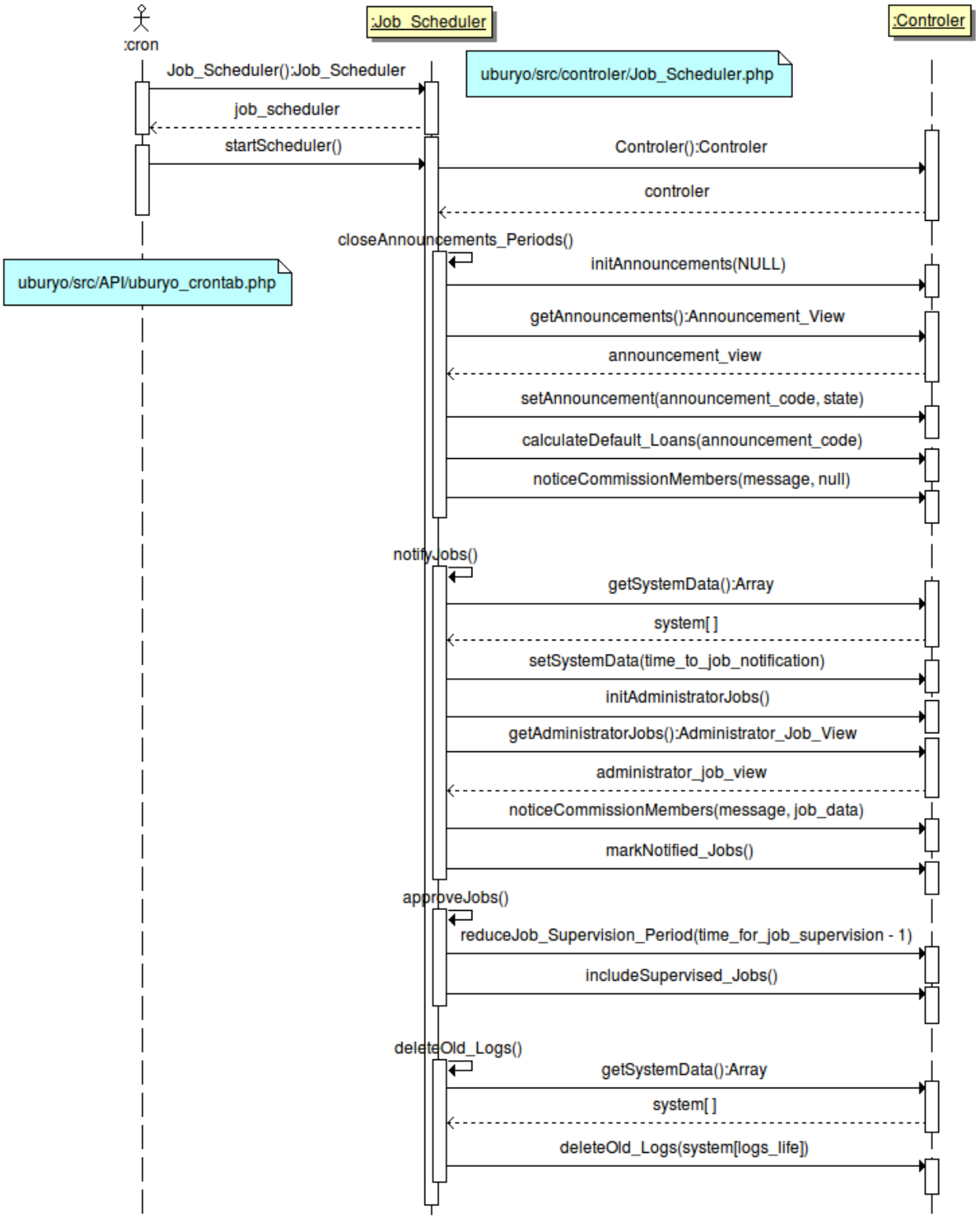


FIGURE 4.7. Job Scheduler Sequence Diagram

CHAPTER 5

DEVELOPMENT

The idea of this section is to familiarize new developers with manners, processes and tools adopted during Uburyo development phase by developers. As open source contributors know is required a big effort to join themselves to any project of this kind. Therefore, this information attempts to make easier this joining period order to get more and more contributors.

5.1. Sourceforge.net

"Sourceforge.net is the world's largest open source software development web site. We provide free services that help people build cool stuff and share it with a global audience. See a list of our software development features.

As of February, 2009, more than 230,000 software projects have been registered to use our services by more than 2 million registered users, making SourceForge.net the largest collection of open source tools and applications on the net. " [SOU10a].

SourceForge.net among others is offering to their users the following services [SOU10b]:

(1) Hosting
   (a) Code Hosting: Code wants to be free. Host your code on our free and public, warm and cozy SVN, Git, Mercurial, Bazaar, or CVS servers.
   (b) Web Hosting: Help yourself to a cup of our Google juice. Host your project or personal website as a high-ranking SourceForge sub-domain including shell access and web traffic analytics.
   (c) Application Hosting: Choose from a dozen available applications including Trac, MediaWiki, WordPress, and more. One click and we will host any or all of them for you.

(2) Development:
   (a) Tracker: We all have issues. Track yours in our new and improved tracker.
   (b) Forums: Hug it out. Collaborate with your community in our new and improved forums.
   (c) Mailing Lists: Hold the snails. Host your mailing lists on our GNU Mailman servers.
   (d) Wiki: We will host a Trac or MediaWiki server for you.
   (e) Blog: If it is worth saying, say it right. We will host a WordPress and/or Laconica server for you.

(3) Distribution:
   (a) File Release: Release your software for easy download on every platform you support. Our system will auto-detect your user's platform to deliver exactly what they need.
   (b) Worldwide Mirror Network: Put your software where the people are. Our download mirror network spans 5 continents so your users get the fastest download available.
   (c) Statistics: Track your download, tracker, forum, and code activity.

TABLE 5.1. Uburyo Development Status Table

Implemented	40
Incomplete	12
Non-implemented	7
Initiated (Implemented + Incomplete)	52
Number of Requirements	59
% Implemented	88,14

(4) Community:
   (a) Exposure: Not THAT kind of exposure. Be seen by our millions of visitors, and on the first page of web search results. The world should know about your software.
   (b) Millions of Users: 2 Million heads are better than one. More users means more bug reports, more patches, and more feedback. Use our "Help Wanted" area to recruit talent for your project.
   (c) Support Staff: Our tireless support staff not only helps you during your development, but helps your users too.
   (d) Open Source Tradition: Over the last 10 years we have helped launch prodigious projects like MySQL, Python, JRuby, JBoss, SugarCRM, and others.

Uburyo development is not using all these offered features but some of them. It is necessary to know how Uburyo developers are using these SourceForge.net features in order to contribute as developer in this project. In the next section are explained all the Uburyo development team processes and how it uses SourceForge.net for getting the open source development model.

5.2. Development processes and useful documents

5.2.1. Managing Software Requirements.

After defining and closing the SRS document (Section 2.1) among users (UNG staff, Application Form), developers and supervisors (TEDECO) a new document called Requirements Management Matrix was created [RM10a]. The goal of this document is to know at any moment the global state of Uburyo development. How much work has been done, which work is being done and how much work is still unresolved. Therefore, this document will be updated constantly and this is the main reason to include it in SourceForge.net CVS..

This document is just a table with all Uburyo requirements included. For each requirement is attached its state {Implemented, incomplete, non-implemented}, some comments about its state and its priority too.

And, at the end of the document also is included a summary table as it is shown in table 5.1.

With this table the development team are able to know what is the state of the project at any time. Data visualized in table 5.1 is real data taken at the date of this document.

5.2.2. Managing Data Model.

Cooperation development requires well established processes and communications in order to get a share knowledge project state among participants. Software data model is something that is constantly changing during software life cycle. In order to go ahead everybody together and notices as soon as possible any model modification the following tools and processes were adopted.

5.2.2.1. *MySQL WorkBench.*

"MySQL Workbench is a cross-platform, visual database design tool developed by MySQL. It is the highly anticipated successor application of the DBDesigner4 project. MySQL Workbench will be available as a native GUI tool on Window, Linux and OS X." [MyS10].

Uburyo development team has adopted this platform [RM10e] for maintaining easier Uburyo data model. It allows to design graphically Entity-Relationship database model, generates database creation scripts and moreover synchronizes it with already created databases. This makes easier to install new versions in servers where Uburyo is already installed.

5.2.2.2. *Data Dictionary.*

Nowadays management software are supported by databases composed by many tables with many columns. Because column names are short they cannot describe the nature of the information that will be stored in it. Developers must know the meaning of each table column in order to avoid information duplication and use that information in the right way.

Uburyo development team uses a spreadsheet managed by the CVS [RM10d] for describing and explaining information nature stored in each database table column.

5.2.2.3. *Data Model Changing Process.*

As readers have noticed, cooperation development implies to share the last knowledge version as soon as it is possible. Knowledge in software is clearly represented by data models as it is easy to deduce software state from them.

For maintaining the last data model version shared among the different development participants Uburyo development team is using, as we explained before, two different documents: Visual Database Design in MySQL Workbench [RM10e] and Data Dictionary [RM10d].

Moreover, every modification done in source code is registered by Uburyo CVS automatically. This is not the same with database modifications, therefore Uburyo development team created a SourceForge.net forum [TED10b] in order to post each database modification performed and also when it was done. Obviously, this forum is only accessible by Uburyo developers.

In this forum is posted each ALTER, CREATE or another modification SQL statement performed by any developer at the moment that modification is being performed. Like this, is really easy for developers adapt their local software versions to the CVS head version by executing last modifications SQL statements needed.

Thus, is very important that each time a developer performs any Uburyo database modification updates the Visual Database Design, Data Dictionary and posts that change in the Database Modifications forum.

5.2.3. Managing Bugs.

5.2.3.1. *Bug Tracking.*

When a pre-alpha software version is got bugs begin to appear. What is a bug?

"A software bug is the common term used to describe an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code. A program that contains a large number of bugs, and/or bugs that seriously interfere with its functionality, is said to be buggy. Reports

detailing bugs in a program are commonly known as bug reports, fault reports, problem reports, trouble reports, change requests, and so forth." [WIK10b].

Therefore, it becomes important to adopt a process for managing bugs when a pre-alpha software version is got. This process is commonly known as Bug Tracking and many tools to support it can be found in the net.

"A bug tracking system is a software application that is designed to help quality assurance and programmers keep track of reported software bugs in their work. It may be regarded as a sort of issue tracking system." [WIK10c].

5.2.3.2. *Mantis.*

"MantisBT is a free popular web-based bug-tracking system (feature list). It is written in the PHP scripting language and works with MySQL, MS SQL, and PostgreSQL databases and a web-server. MantisBT has been installed on Windows, Linux, Mac OS, OS/2, and others. Almost any web browser should be able to function as a client. It is released under the terms of the GNU General Public License (GPL)." [MAN10].

Uburyo development team adopted this system for tracking its bugs and also integrated it with SourceForge.net [TED10a]. Here anyone may report bugs but just developers could manage them.

CHAPTER 6

TESTS

The importance of software testing and its impact on software cannot be underestimated. Software testing is a fundamental component of software quality assurance and it represents a review of specification, design and coding. The greater visibility of software systems and the cost associated with software failure are motivating factors for planning, through testing.[FIN09].

During the process we designed and executed some test to check different functions and elements that forms the software. But we are not going to paste here those tests. In this chapter we are going to describe a complete test plan that can help to anybody to decide if a new version is fully usable and can be released as a new version.

We prefer to describe this plan like a checklist in order to make this bored (but very important) task easier and more interesting. In some points we introduce most common problems that could appear and their possible causes.

This checklist is divided in four important parts: The fully process, upload/download files, emails and audit. If the system is going to be tested after adding a new feature, it is important to check these four parts and also the new feature.

6.1. Fully process

In this part we are going to test the greatest number of features simulating a complete process of an announcement's life cycle. In each step appears the name of the role that performs the action. For a fully description of how to do the different actions go to the specific role guide.

(1) [Administrator] Create a new announcement for testing. If you can not log in Uburyo is possible that database properties were not be configured (review <install_path>/uburyo/src/CONF/db_properties.php file).
(2) [Administrator] Configure the system and check, along announcement's life cycle, if software behaves as parameters say. Obviously is necessary to know and understand how process and software works and how influence each parameter during announcement's life cycle.
(3) [Administrator] Create at least one Administrative user, two Commission Member users and one Administrator user more. Check that they appear in the users list.
(4) [Administrator] Change password of one of this new users. Check (by log out and log in) that password has been well changed.
(5) [Student] Fill up at least two application forms in the new announcement created in the step 1.
(6) [Administrator] Close the application period for the announcement. Check that the status change and students cannot apply any more in that announcement.
(7) [Administrative] Review two introduced applications, change some data and validate the application. Check that the status and data of the application change.

(8) [Administrator] Close the validating period for the announcement. Check that the status change and administratives cannot validate any application in that announcement.
(9) [Commission member] With both commission member created in the step 2 evaluate that two application created and validated in this new announcement. Check that the evaluations have been stored with their corresponding comments and they are visible by all commission members.
(10) [Administrator] Close the evaluation period for the announcement. Check that the status change and also the automatic ordered list is well generated.
(11) [Commission member] Review the amounts generates automatically by the system and comment the assignation. Check that the comment is stored in the system and readable for the other commission members.
(12) [Administrator] Read comments about the assignations and redistribute quantities assigned. Finally, close the assignment period. Check that changes were stored in the system and the status of the announcement changes.
(13) [Administrative] Check that can read the Loan Resolution.
(14) [Administrative] Introduce at least two new jobs. Check that this jobs was stored in the system.
(15) [Commission member] Check that new introduced jobs in the last step appears in the jobs list and reject one of them. Check that it appears as rejected in the list.
(16) [Administrator] Recover the rejected job and approved two jobs. Check that the status change.
(17) [Student] Check that approved jobs appear for all that visit Uburyo web.
(18) [Administrative] Assign a job to some student.
(19) [Administrator] Check that status of the assigned job changed and it does not appear any more inside the employment bureau.
(20) [Administrative] Modify the debt and the profits and check if changes were registered. Finalize any assigned job and verify its correctly closing. Finally close the debt and check that it was well closed.

6.2. Upload and download files

We are going to check special tasks that allow to different roles upload and download files from the system.

(1) [Student] From the main page, download the different documents available (available in the default installation). Check that can be read.
(2) [Administrative] Upload some files and check that can be downloaded and are the correct files. If the upload is impossible check the permission of the <install_path>/uburyo/doc folder and also UBURYO_PATH constant in <install_path> /uburyo/src/CONF/uburyo_conf.php file.

6.3. Email

We are going to check the mailer utility incorporate inside Uburyo.

(1) [Administrator] Send an email with the emailer utility to the administrator group. Check if the email reached its destination. If there are any error or the email does not reach its destination, test if the phpMailer configuration is correct in file <install_path>/uburyo/src/CONF/uburyo_conf.php and you have Internet connection.

(2) [Administrator] After a Fully process test (if email parameter was enabled) check in commission members in-boxes if there are the corresponding emails. If not, check if static table email is loaded with corresponding emails.

6.4. audit

We are going to test the audit part.

(1) [Administrator] Configure the system to audit different actions. If actions do not appear check if static table action was loaded.
(2) [Any role] Do the different actions configured to be audited, scoring when you do what.
(3) [Administrator] Check that the notes written in the last step match with the system log. If all the notes match, but the time and date do not match, can be a problem with the server date and time configuration. Let's check Uburyo system configuration interface for verifying date and time are correct.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK LINES

It is early to obtain well-founded conclusions of the project and its performance in a real situation. But, despite as the time goes by will be got more data, it is possible to offer some first conclusions at the date of the document.

7.1. ICT and Free Software in Cooperation for Development

The authors of this work were looking for different projects to finish their studies of Computer Science. Finally, they decided to merge two different realities that are not very common in their area: Cooperation for Development and Free Software.

In the last decade there has been a large increase in the presence of ICT in development projects. In most cases this presence is just necessary tools to complete the project: from radio stations to informative websites, going through some more creative and complex uses as Hispanic-American Health Link[EHA08] (EHAS for its Spanish abbreviation *Enlace Hispano-Americano de Salud*). This project uses different ICT to improve health care processes.

In addition to this increase, the number of cooperation projects whose main objective is closer to the field of ICT than another field is increasing, being not longer just tools to be something else. In these projects, technology is not as important as contents or services. In this line appear projects like Digital Doorways[Mer10] which aim is to provide people in rural and disadvantaged areas with freely accessible computer equipment and open source software, enabling them to experiment and learn without formal training and with minimal external input. Other project of this kind is the Ushahidi Platform[Ush10] that allows anyone to gather distributed data via SMS, email or web and visualize it on a map or timeline. Their goal is to create the simplest way of aggregating information from the public for use in crisis response.

But during the last years, in Cooperation for Development projects environments, is very frequent to find Information and Communication Technologies understood as tools to strengthen projects. Many people and organizations tend to refuse pure ICT cooperation projects, arguing that is not a basic need. And inside ICT projects, software-based projects are the last in priority.

The authors of this work think that pure ICT projects are also a good chance because we live in a globalized world, where information and knowledge have more importance each day, and developing countries could use it in order to develop themselves.

In the other side is the development of a free software project. There are many important and special characteristics that surround a Free Software project and converts it in a special project: use of tools for cooperation development, what kind of license choose, a documented code and a very clear English documentation for the following developers, etc.

Inside this two realities we have some positive experiences and other negatives. Here are exposed the positive experiences:

TABLE 7.1. Uburyo Account State

UNG Profits	1249$
UNG Saves	0$
UNG Added Value	486$
To be paid	452 $
TOTAL LOANS	2187$

- Thanks to Free Software and its working philosophy any person can be incorporated to any phase of the development easily. The best example is Gerard MANIRAKIZA, a student with an Uburyo loan that performs an Uburyo job consisting in doing local support of Uburyo software in the last phase of development.
- The use of a software forge as a fundamental development tool. It does the version control of the software and documents, provides tools for the communication among developers (forum and bug tracking), to publish the status of the project (main web, wiki, news...), etc.
- During deployment and from the beginning, software interaction was performed just by local staff and not by us, of course always with our training and supervision. This is the best way to do a complete technology transfer.
- The need of create a very simple and easy interface and simplify tasks performed by different roles, thinking always in people with little or none computer experience.

As negative conclusions, we have two main points that must be improved during next experiences:

- Not having a student or local staff integrated as a developer or viewer from the project beginning. We think that this point would help technology transfer and also to detect some problems (that now only use can show) like the problem with the mandatory fields in the application.
- Do more advertising of the project, especially oriented to university staff. It is very important that university employers know the Uburyo system and how they can help proposing jobs with which the student can return the loan. In this first experience we prepared a presentation, some letters and a general guide, but we think that is not enough, specially in this task (propose jobs) that is essential to all the process. It is completely necessary to do a human-contact advertisement effort.

7.2. Experience at the UNG

In this first system deployment at the University of Ngozi we have obtained, nowadays, very good results. We got a lot of loan applications and finally, nine students got a loan amounting to the registration fees (243$ per student, a total of 2187$). At the same time, all these nine students obtained different jobs to reduce their debts. It must be told that an external company proposed some of these jobs. Therefore, we got the more complex kind of jobs at first Uburyo announcement, which will allow the system to reach sustainability faster.

Specifically, six of the nine jobs proposed to date are from this external company and allow direct entry of money to the bag of loans (UNG Profits row in the Table 7.1). Because these jobs pay for worked hours and not all the students give the same number of hours, not all the students earn enough money to cancel their debts (To be paid row). The

other three jobs give to he University an Added Value but do not save money to it (UNG Saves and UNG Added Value rows). One of these three students do not finish yet her job and the quantity was added to the To be paid row.

If we define, as a simple index of sustainability, the percentage of money returned, we can see that this first experience has a fairly high sustainability index (79,3%). Is important to emphasize that the experience has not yet been terminated, so this percentage may increase or decrease, but anyway, it is a very high percentage for the first year.

Students with loans expressed their happiness with the given opportunity and jobs they are doing to return the loan. They are showing that they will be great professionals in future jobs and they are getting an important professional experience before finishing their studies. This is not a common practice in developing countries.

This professional experience is specially important for female students, that before finishing their studies, they are doing jobs traditionally reserved for men. Is a very little step to fight against sex roles prejudices and female students are very proud.

Other good sense from Ngozi is that the company that propose the most of the jobs are thinking about hiring some of the students that have worked with it. It is a very good consequence that exceed the objectives of the project.

The university director and his team are very interesting in the project and how it evolves along the time. This point added to the complete technology transference assure that the project will continue. One of the tasks that most worry them is the jobs generation. The accounting department is the most skeptic, obviously, because it lives in first person the disaster of other scholarship and loan programs, and because the introduction of a new process and software is for them an extra effort. It is a task of the educational center management encourages their active participation in the project, compensating and rewarding their effort.

It is important to talk about one special department of the university that has collaborated very close with Uburyo: the responsible of generate new services and look for financing, what we call in developed countries: Research and Development department. They got the contact with the company that has offered six of the nine jobs done by students today.

At the publication date of this report, a second announcement is in progress. In these new experience the loans are offered to Computer Science and Interpreting/Translation students, one university faculty more than during the first announcement. A total of 22 loans will be given through the obtained results and the Solidaridanza support.

7.3. Work summary

The present work, as it has been said before, has been developed by the coordinated effort of two Computer Science students: Máximo Ramirez y Eduardo Martinez-Larraz.

This work not only consists of developing source code, but also the compilation of the characteristics of free software and the requirements necessary for a cooperation project. Documentation is specially important for its understanding and reuse. The following list is the project material generated and the implication of each author in their generation. Both authors have participated in some way in creating the material, but we will indicate the majority ownership:

- User Requirements Document (just in Spanish): realized through on-site meetings between Maximo Ramirez and Eduardo Martinez-Larraz.
- Software Requirements Specification: realized through on-site meetings between Maximo Ramirez and Eduardo Martinez-Larraz.

- Management Requirements Matrix: realized through on-site meetings between Maximo Ramirez and Eduardo Martinez-Larraz.
- Uburyo Data Model: realized through on-site meetings between Maximo Ramirez and Eduardo Martinez-Larraz.
- Data dictionary: realized through on-site meetings between Maximo Ramirez and Eduardo Martinez-Larraz.
- Source code: realized through on-line coordinated efforts of Maximo Ramirez and Eduardo Martinez-Larraz.
- Backup script: by Eduardo Martinez-Larraz.
- Graphic design: by Eduardo Martinez-Larraz.
- Final Report: realized through on-line coordinated efforts of Maximo Ramirez and Eduardo Martinez-Larraz.
- Manuals and Guides
  - Process Description Manual: by Eduardo Martinez-Larraz.
  - Student Manual: by Maximo Ramirez.
  - Applicator Manual: by Maximo Ramirez.
  - Administrative Manual: by Maximo Ramirez.
  - Commission Member Manual: by Maximo Ramirez.
  - System Administrator Manual: by Maximo Ramirez.
  - Installation Guide: by Maximo Ramirez.
- Templates:
  - Register job form: by Eduardo Martinez-Larraz.
  - Finish job form: by Eduardo Martinez-Larraz.
  - Ended job report index: by Eduardo Martinez-Larraz.
- Created Job Templates report: by Maximo Ramirez.
- Contracts and Commitment letters: by Maximo Ramirez.
- Presentations:
  - Uburyo pre-deployment presentation: by Eduardo Martinez-Larraz.
  - Uburyo presentation for Solidaridanza: by Eduardo Martinez-Larraz.
  - Poster for III International Meeting on ICT for Development Cooperation(just): by Eduardo Martinez-Larraz.
- Uburyo project official web:
  - current version[RM10b]: by Eduardo Martinez-Larraz.
  - new version[RM10c]: by Maximo Ramirez.

7.4. Future work lines

At the end of the project we think that there are some work lines that can be explore to expand Uburyo project:

- API for external applications: There are some situations where is possible that we need a communication between Uburyo and other applications. For example, when we think that Uburyo manages all about the loans, but do not manage money. Each educational center can use a different software to manage its countability, and the money from Uburyo must be managed with this tool. But can be important to merge these two tools (Uburyo and the countability tool). For this case it is necessary to develop an API that allows the communication between both. Other example is when we have an enrollment software and we prefer that Uburyo obtain the student data from this external software. It is a good way to reduce the time of the student application and better to verify these data.

- Uburyo stored a lot of data about the process that supports. These data can be converted in useful information, through reports, statistics and graphics. This information can be used to perform a serious evaluation of the system or to obtain new financial partners.
- Other future work its to create a new role for financial partners, to allow them to see in real time how is working Uburyo. This real-time supervision involves more financial partners with the project and also concrete students.
- To finish, it could be interesting to find any way, to show possible external companies, jobs that students are doing and how they are doing them for other companies. By this work, more companies could collaborate with the project offering new jobs for the students.

CHAPTER 8

BIBLIOGRAPHY

Referencies

[APA10] APACHE. Apache official web page. http://httpd.apache.org/, June 2010. 1.5.4

[BUR92] S. BURBECK. Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc). http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html, January 1992. 3

[COO10] COOP2.0. Iii international meeting on ict for development cooperation. http://encuentro2010.fundacionctic.org/, June 2010. 10.3

[CUR10] CURL. Curl official web page. http://curl.haxx.se/, June 2010. 3.1.3

[CVS10] CVS. Cvs official web page. http://www.cvshome.org/, June 2010. 1.5.4

[ECL10] ECLIPSE. Eclipse offcial web page. http://www.eclipse.org/, June 2010. 1.5.4

[EEL06] P. EELES. Ibm developerworks explanation of software architecture. http://www.ibm.com/developerworks/rational/library/feb06/eeles/, February 2006. 3, 3, 3

[EHA08] EHAS. Enlace hipano americano de salud. http://www.ehas.org/, 2008. 7.1

[FIN09] J. E. FINLANDER. The importance of software testing. http://www.scribd.com/doc/13961163/The-Importance-of-Software-Testing, 4 2009. 6

[GNO10] GNOME. Dia official web page. http://live.gnome.org/Dia, June 2010. 1.5.4

[GNU10a] GNU. The free software definition. http://www.gnu.org/philosophy/free-sw.html, June 2010. 1.5.2

[GNU10b] GNU. Gettext - gnu project. http://www.gnu.org/software/gettext/, June 2010. 4.2

[GTK09] GTK+. Gtk+ (gimp tool kit) project. http://www.gtk.org/, 2009. 1.5.4

[LaT08] LaTeX. Latex project. http://www.latex-project.org/, 2008. 1.5.4

[LYX10] LYX. Lyx official web page. http://www.lyx.org/, June 2010. 1.5.4

[MAN10] MANTIS. Mantis official web page. http://www.mantisbt.org/, June 2010. 1.5.4, 5.2.3.2

[MAR04] T. MARSTON. The model-view-controller (mvc) design pattern for php. http://www.tonymarston.net/php-mysql/model-view-controller.html, May 2004. (document), 3.1, 3.1.4.2

[MED10] MEDIAWIKI. Mediawiki official web page. http://www.mediawiki.org/wiki/MediaWiki, June 2010. 1.5.4

[Mer10] Institute Meraka. Digital doorways. http://www.digitaldoorway.org.za/index_main.php, 3 2010. 7.1

[MyS10] MySQL. Mysql workbench. http://wb.mysql.com/?page_id=6, June 2010. 1.5.4, 5.2.2.1

[NOR08] NORAD. Logical framework approach offical manual (from norwegian agency for development cooperation). http://www.scribd.com/doc/

2982757/Manual-del-Enfoque-de-Marco-Logico, May 2008. 1.1.2

[ORA10a] ORACLE. Mysql official web page. http://www.mysql.com/, June 2010. 1.5.4

[ORA10b] ORACLE. Open office official web page. http://www.openoffice.org/, June 2010. 1.5.4

[PAR10] PARC. Palo alto research cernter official web page. http://www.parc.com/about/, June 2010. 3

[PHP09] PHPGettext. php-gettext project web. https://launchpad.net/php-gettext/, June 2009. 4.2.1

[PHP10] PHPBB. Phpbb official web page. http://www.phpbb.com/, June 2010. 1.5.4

[PZ05] K. PÉREZ DE ARMIÑO and N. ZABALA. Appropiate technology definition. http://dicc.hegoa.efaber.net/listar/mostrar/214, January 2005. 1.5.1

[RAM10] M. RAMÍREZ ROBLES. Uburyo volume ii: Delivering of a sustainable system of loans for education. Master's thesis, Facultad de Informática, Universidad Politécnica de Madrid, November 2010. 1.6, 2.1, 2.2, 2.2, 4.2.1, 4.3

[REE10] T. REENSKAUG. Personal web page of trygve reenskaug. http://heim.ifi.uio.no/~trygver/, June 2010. 3

[RM10a] M. RAMÍREZ ROBLES and E. MARTINEZ LARRAZ. Requirements management matrix. http://uburyo.cvs.sourceforge.net/viewvc/*checkout*/uburyo/uburyo/doc/requirements_management_matrix.ods, June 2010. 2.2, 5.2.1

[RM10b] M. RAMÍREZ ROBLES and E. MARTINEZ LARRAZ. Uburyo current website. http://uburyo.sf.net, June 2010. 7.3

[RM10c] M. RAMÍREZ ROBLES and E. MARTINEZ LARRAZ. Uburyo new website. http://uburyo.sf.net/new, June 2010. 7.3

[RM10d] M. RAMÍREZ ROBLES and E. MARTINEZ LARRAZ. Uburyo's data dictionary. http://uburyo.cvs.sourceforge.net/viewvc/*checkout*/uburyo/uburyo/doc/data_dictionary.xls, June 2010. 5.2.2.2, 5.2.2.3

[RM10e] M. RAMÍREZ ROBLES and E. MARTINEZ LARRAZ. Uburyo's data model (mwb) - mysql workbench. http://uburyo.cvs.sourceforge.net/viewvc/*checkout*/uburyo/uburyo/doc/uburyo_data_model.mwb, June 2010. 5.2.2.1, 5.2.2.3

[SAT07] SATYAM. Internacionalización y localización: usando php-gettext. http://www.satyam.com.ar/blog/2007/01/16/internacionalizacion-y-localizacion-usando-php-gettext/, January 2007. 4.2.1

[SCH73] E. F. SCHUMACHER. Small is beautiful, 1973. 1.5.1

[SOL10] SOLIDARIDANZA. Solidaridanza. http://www.solidaridanza.com, June 2010. 10.2

[SOU10a] SOURCEFORGE. About sourceforge.net. https://sourceforge.net/about, June 2010. 1.5.4, 5.1

[SOU10b] SOURCEFORGE. Sourceforge.net features. https://sourceforge.net/register-project/features.php, June 2010. 5.1

[SUN02] SUN. Java blueprints model-view-controler. http://java.sun.com/blueprints/patterns/MVC-detailed.html, January 2002. (document), 3, 3.2

[TED06] TEDECO. Teson project. http://tedeco.fi.upm.es/proyectos/teson, 2006. 1.1.1.1, 1.1.2

[TED08] TEDECO. Ticamen project. http://tedeco.fi.upm.es/proyectos/ticamen, 2008. 1.1, 1.1.1.2, 1.1.2

[TED10a] TEDECO. Uburyo's bug tracking system - mantis at sourceforge.net. https://sourceforge.net/apps/mantisbt/uburyo/my_view_page.php, June 2010. 2.2, 5.2.3.2

[TED10b] TEDECO. Uburyo's database modifications forum. https://sourceforge.net/projects/uburyo/forums/forum/898375/topic/3681085, June 2010. 5.2.2.3

[UBU10] UBUNTU. Ubuntu official web page. http://www.ubuntu.com/, June 2010. 1.5.4

[Ush10] Ushahidi. Ushahidi. http://www.ushahidi.com/, 3 2010. 7.1

[WIK09a] WIKIPEDIA. Internationalization and localization. http://en.wikipedia.org/wiki/Internationalization_and_localization, June 2009. 4.2.1

[WIK09b] WIKIPEDIA. Logical framework approach definition. http://en.wikipedia.org/wiki/Logical_framework_approach, 2009. 1.1.2

[WIK10a] WIKIPEDIA. Appropiate technology definition. http://en.wikipedia.org/wiki/Appropriate_technology, June 2010. 1.5.1

[WIK10b] WIKIPEDIA. Bug definition. http://en.wikipedia.org/wiki/Software_bug, June 2010. 5.2.3.1

[WIK10c] WIKIPEDIA. Bug tracking definition. http://en.wikipedia.org/wiki/Bug_tracking_system, June 2010. 5.2.3.1

[WIK10d] WIKIPEDIA. Cron definition. http://en.wikipedia.org/wiki/Cron, June 2010. 4.3

[WIK10e] WIKIPEDIA. Evolutionary prototyping. http://en.wikipedia.org/wiki/Software_prototyping, June 2010. 2.1.3.6

[WIK10f] WIKIPEDIA. Microsoft windows task scheduler. http://en.wikipedia.org/wiki/Task_Scheduler, June 2010. 4.3

[WIK10g] WIKIPEDIA. Requirements elicitation process definition. http://en.wikipedia.org/wiki/Requirements_elicitation, June 2010. 2.2

[WIK10h] WIKIPEDIA. Scholarship definition. http://en.wikipedia.org/wiki/Scholarship, June 2010. 1.2

[WIL04] P. WILLIAM LOUNT. What is smalltalk. http://www.smalltalk.org/smalltalk/whatissmalltalk.html, August 2004. 3

Part 2

SECOND PART: APPENDIXES

CHAPTER 9

FORMS

Here are attached the last application form used by UNG for getting student data in order to assign its scholarships following the last non-sustainable system. Student Application Form in Uburyo is strongly based on it.

CONVENTION TRIPARTITE ENTRE L'ETUDIANT BENEFICIAIRE DU CREDIT-ETUDIANT, L'UNIVERSITE DE NGOZI ET L'AVALISEUR

Entre les soussignés.

1 L'étudiant
contractant de première part, ci-après dénommé « L'EMPRUNTEUR »
2 L'Université de Ngozi ayant son siège social à NGOZI, B.P. 137 NGOZI, représentée par Mgr Stanislas KABURUNGU, Recteur de l'Université de Ngozi
contractant de deuxième part, ci-après dénommée « UNIVERSITE DE NGOZI »
3 contractant de troisième part, ci-après dénommé « L'AVALISEUR »

Il a été convenu ce qui suit

Article 1 L'Université de Ngozi accorde à Mme, Mlle, M qui en fait la demande, une autorisation de crédit-étudiant de FBU couvrant les frais d'inscription au rôle et aux cours pour l'(les) année(s) académique(s) et de FBU complétant les frais de subsistance soit un total de FBU

Article 2 Madame Mademoiselle Monsieur
l'étudiant s'engage à
- suivre avec assiduité les enseignements théoriques et pratiques dispensés par l'Université de NGOZI dans la filière de formation de son choix
- passer tous les tests et examens prévus par le règlement académique ;
- **fournir avec exactitude et dans les délais prévus tout document administratif ou académique lui demandé par la COFIDE et l'Avaliseur;**
- signaler dès son entrée dans la vie professionnelle, le nom et l'adresse de son employeur à l'Université de NGOZI et à l'Avaliseur
- honorer les modalités de remboursement du crédit convenues avec la COFIDE

Article 3 L'Université de Ngozi s'engage à
- assurer une formation scientifique et humaine à l'étudiant
- Conserver le diplôme du lauréat jusqu'au remboursement total du crédit

Article 4 L'étudiant s'engage à rembourser par mensualités le crédit dû y compris les intérêts, commissions et frais tels qu'arrêtés à la date de l'interruption des études quelles qu'en soient les raisons
Le remboursement s'effectuera à la COFIDE pour une durée correspondant à celle des études

Article 5 En cas de défaillance de l'emprunteur l'avaliseur s'engage à rembourser la totalité du montant dû capital et intérêt commission et frais

Article 6 Pour la bonne fin de l'opération l'avaliseur réalise déjà un dépôt de 20% du crédit cautionné. **La quittance du dépôt porte le numéro.**

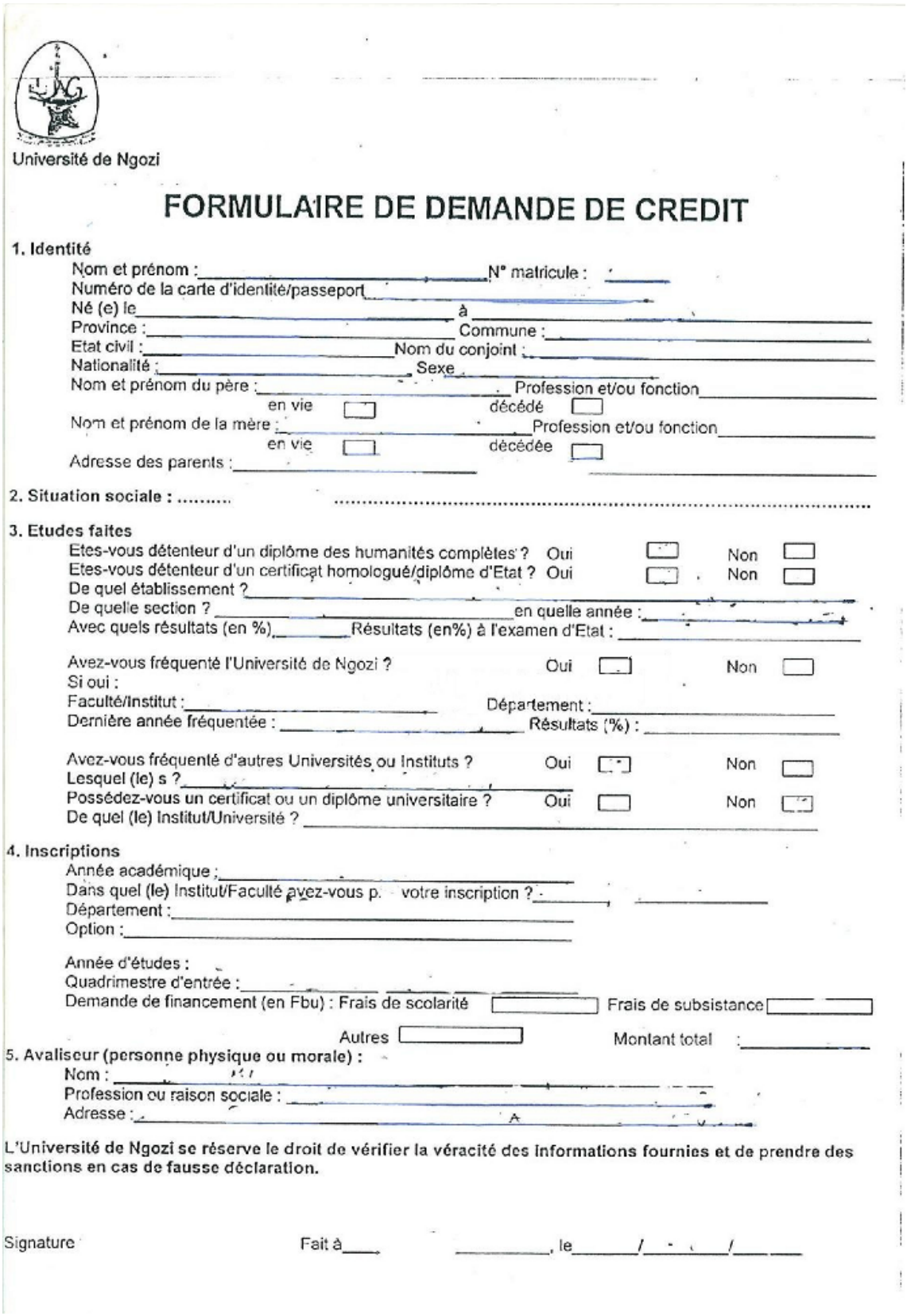
Article 7 Tout litige qui surviendra entre les parties concernées sera réglé par voie d'arbitrage. A défaut d'entente il sera porté devant les juridictions du Burundi

Ecrire au stylo avant le nom et la signature « **Lu et approuvé** »

Fait à Ngozi le :200

L'Emprunteur L'Avaliseur **L'Université de Ngozi**

FIGURE 9.1. UNG Student Application Form (Page 1)



Université de Ngozi

FORMULAIRE DE DEMANDE DE CREDIT

1. Identité

Nom et prénom : _____ N° matricule : _____
Numéro de la carte d'identité/passeport _____
Né (e) le _____ à _____
Province : _____ Commune : _____
Etat civil : _____ Nom du conjoint : _____
Nationalité : _____ Sexe _____
Nom et prénom du père : _____ Profession et/ou fonction _____
en vie ☐ décédé ☐
Nom et prénom de la mère : _____ Profession et/ou fonction _____
en vie ☐ décédée ☐
Adresse des parents : _____

2. Situation sociale : .....

3. Etudes faites

Etes-vous détenteur d'un diplôme des humanités complètes ? Oui ☐ Non ☐
Etes-vous détenteur d'un certificat homologué/diplôme d'Etat ? Oui ☐ Non ☐
De quel établissement ? _____
De quelle section ? _____ en quelle année : _____
Avec quels résultats (en %) _____ Résultats (en%) à l'examen d'Etat : _____

Avez-vous fréquenté l'Université de Ngozi ? Oui ☐ Non ☐
Si oui :
Faculté/Institut : _____ Département : _____
Dernière année fréquentée : _____ Résultats (%) : _____

Avez-vous fréquenté d'autres Universités ou Instituts ? Oui ☐ Non ☐
Lesquel (le) s ? _____
Possédez-vous un certificat ou un diplôme universitaire ? Oui ☐ Non ☐
De quel (le) Institut/Université ? _____

4. Inscriptions

Année académique : _____
Dans quel (le) Institut/Faculté avez-vous p. votre inscription ? _____
Département : _____
Option : _____

Année d'études :
Quadrimestre d'entrée : _____
Demande de financement (en Fbu) : Frais de scolarité ☐ Frais de subsistance ☐
Autres ☐ Montant total : _____

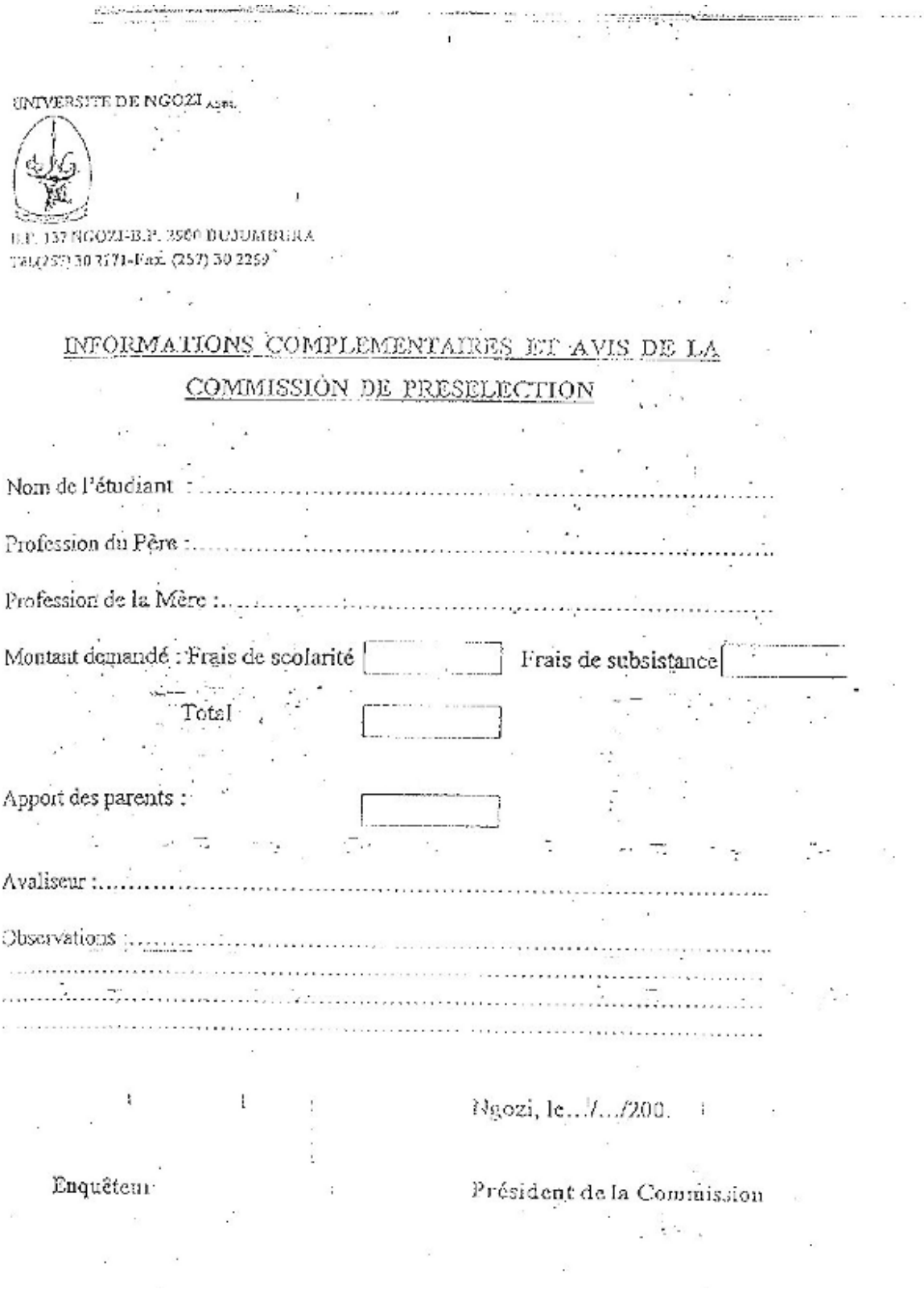
5. Avaliseur (personne physique ou morale) :

Nom : _____
Profession ou raison sociale : _____
Adresse : _____

L'Université de Ngozi se réserve le droit de vérifier la véracité des informations fournies et de prendre des sanctions en cas de fausse déclaration.

Signature Fait à _____, le _____/_____/_____

FIGURE 9.2. UNG Student Application Form (Page 2)



UNIVERSITE DE NGOZI Asbl.

B.P. 137 NGOZI-B.P. 2500 BUJUMBURA
Tel.(257) 30 2171-Fax. (257) 30 2259

INFORMATIONS COMPLEMENTAIRES ET AVIS DE LA COMMISSION DE PRESELECTION

Nom de l'étudiant : .....

Profession du Père : .....

Profession de la Mère : .....

Montant demandé : Frais de scolarité [ ] Frais de subsistance [ ]

Total [ ]

Apport des parents : [ ]

Avaliseur : .....

Observations : .....
.....
.....
.....

Ngozi, le.../.../200.

Enquêteur

Président de la Commission

FIGURE 9.3. UNG Student Application Form (Page 3)

CHAPTER 10

PRESENTATIONS

10.1. Uburyo Pre-Deployment Presentation

The following presentation was created to make in the University of Ngozi before the deployment of the software. The objective is to explain the system and motivate people of UNG to participate in the project. This was shown in the UNG a couple of months before the deployment started. With the slides there is a guide to show it.

10.1.1. Presentation Guide.

10.1.1.1. *SLIDE 1 [Figure 10.1]: FRONT.*

It is the moment to explain why this project is called Uburyo (=Opportunity): because we believe that it is a tool that offer a lot of opportunities:

- To the students offer the opportunity to study
- To the university offer the opportunity to continue growing and doing the great work they do.

10.1.1.2. *SLIDE 2 [Figure 10.2]: INDEX..*

The index of the presentation. Nothing special except the picture that is the main page of the Uburyo software.

FIGURE 10.1. Pre-Deployment Presentation slide 1

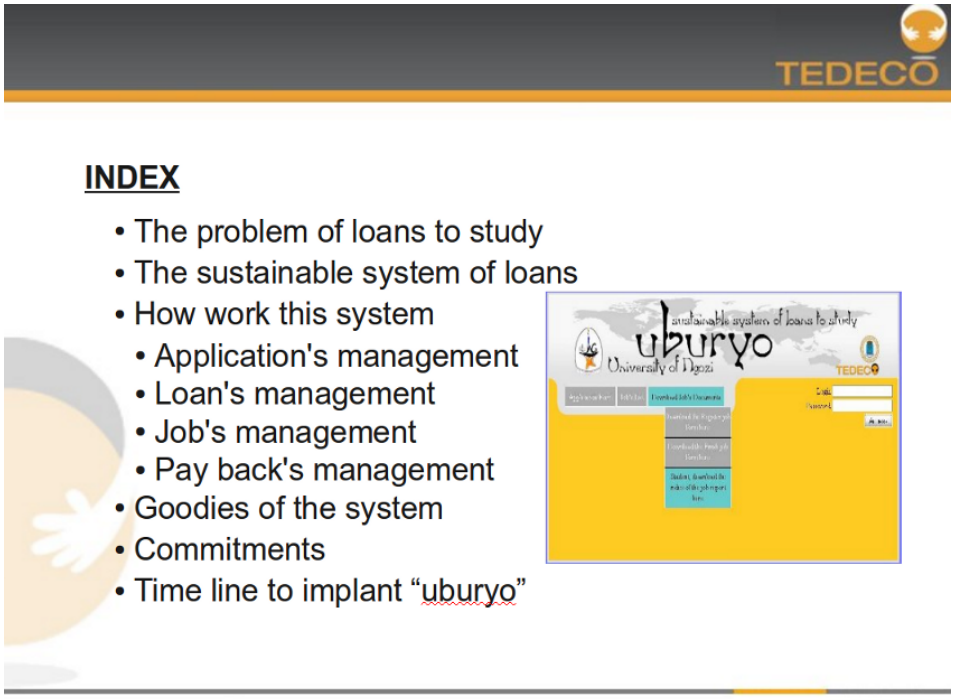


FIGURE 10.2. Pre-Deployment Presentation slide 2

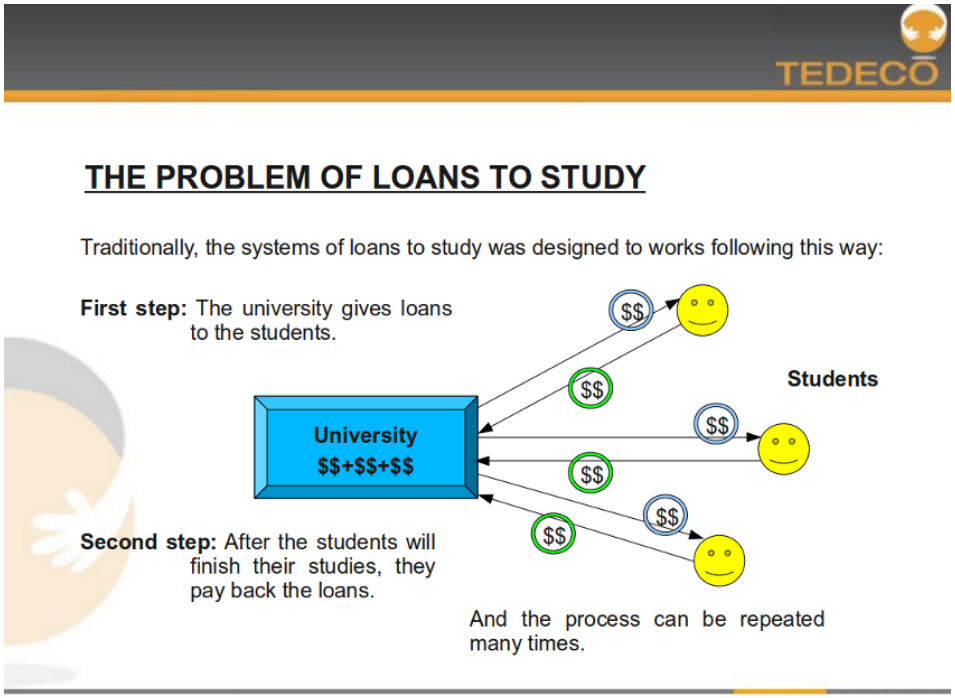


FIGURE 10.3. Pre-Deployment Presentation slide 3

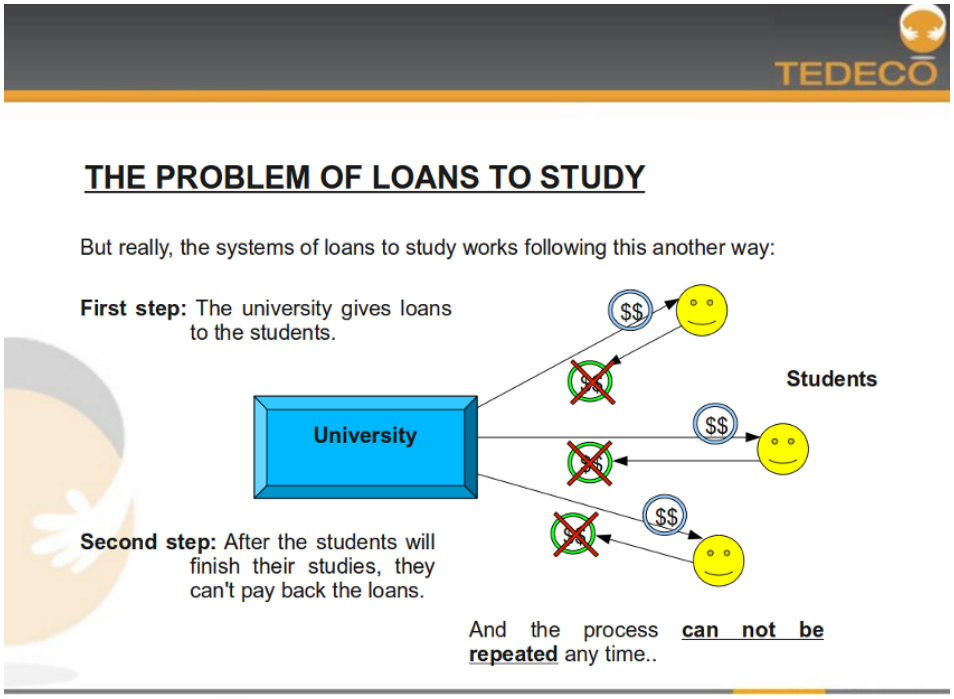


FIGURE 10.4. Pre-Deployment Presentation slide 4

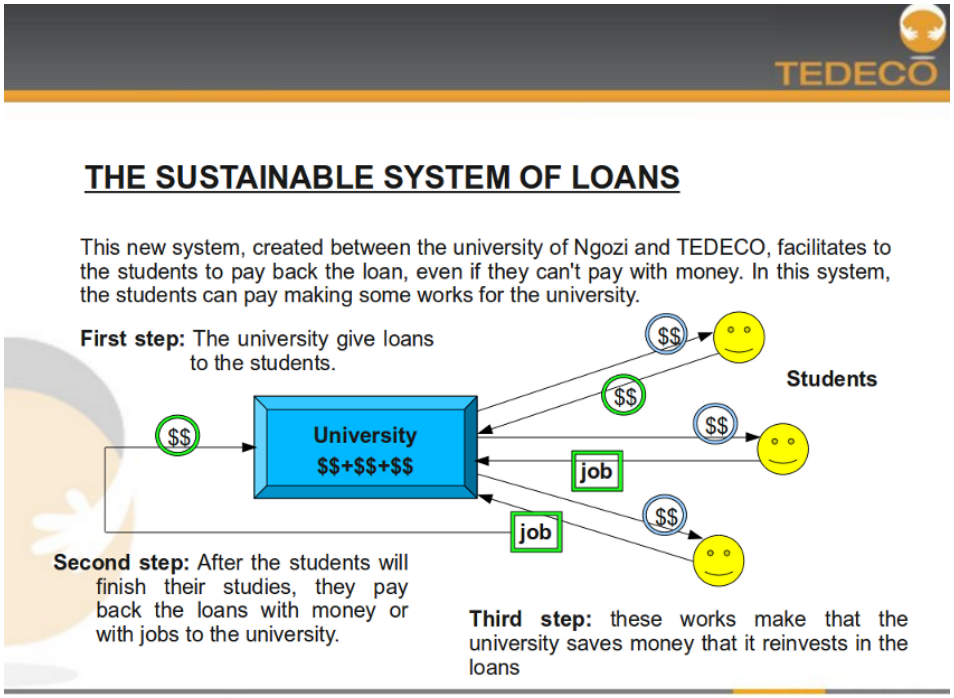


FIGURE 10.5. Pre-Deployment Presentation slide 5

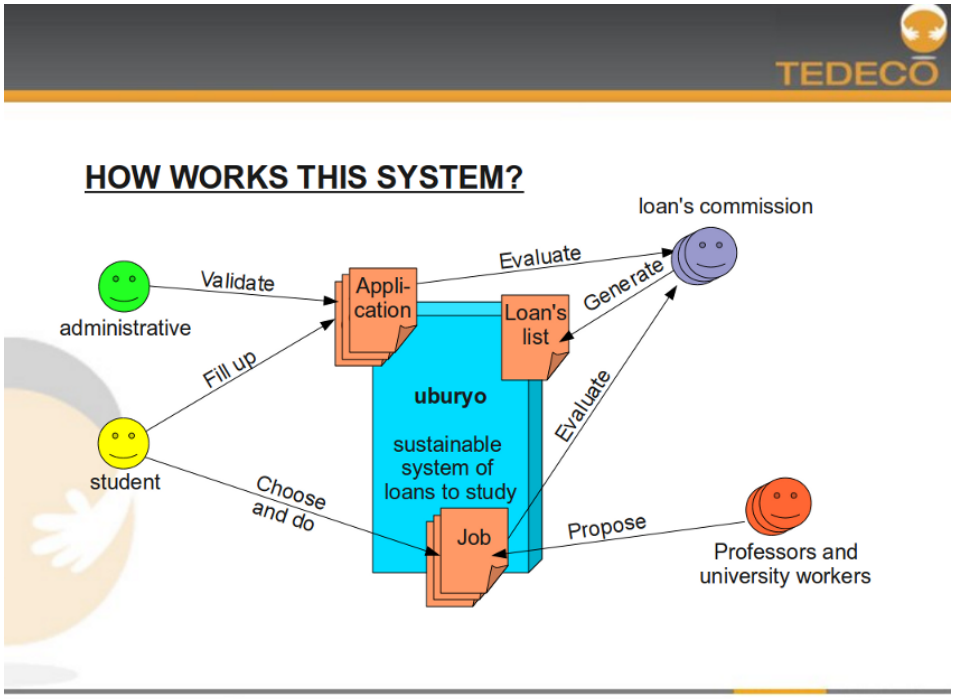


FIGURE 10.6. Pre-Deployment Presentation slide 6

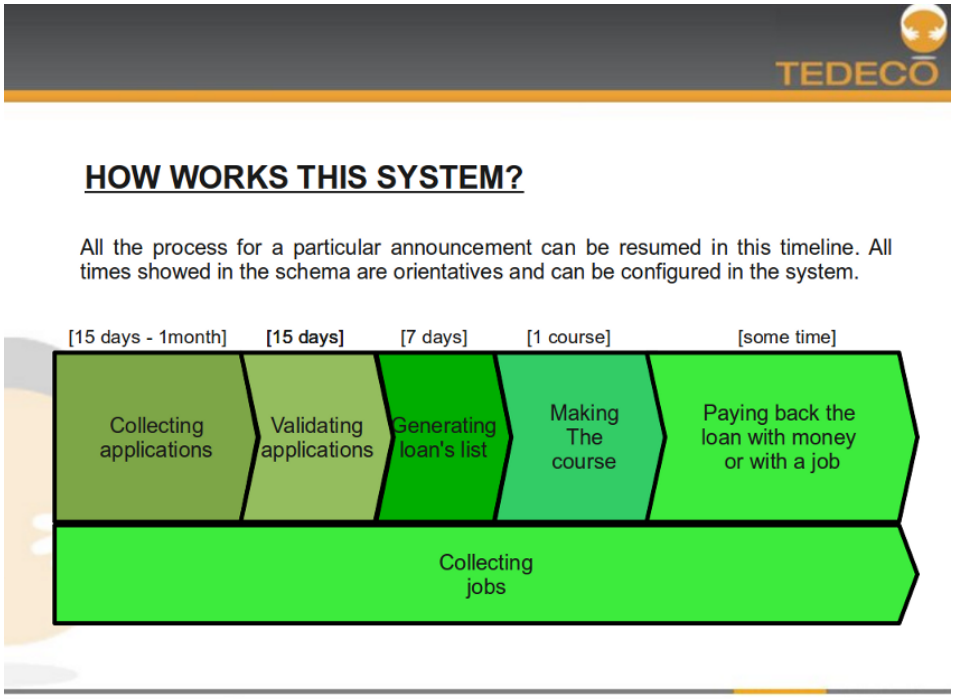


FIGURE 10.7. Pre-Deployment Presentation slide 7

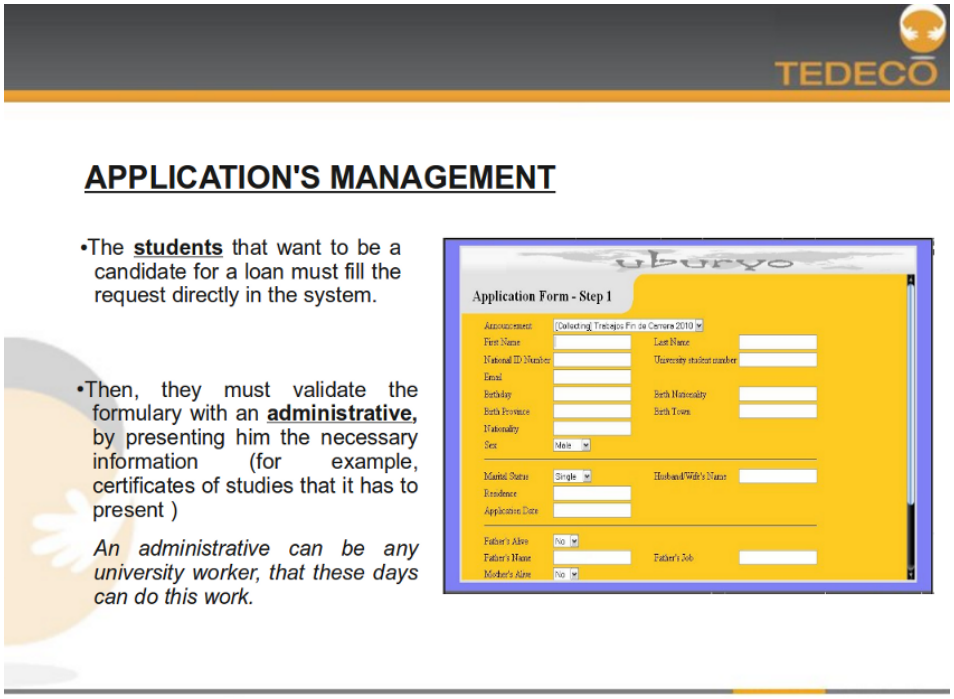


FIGURE 10.8. Pre-Deployment Presentation slide 8

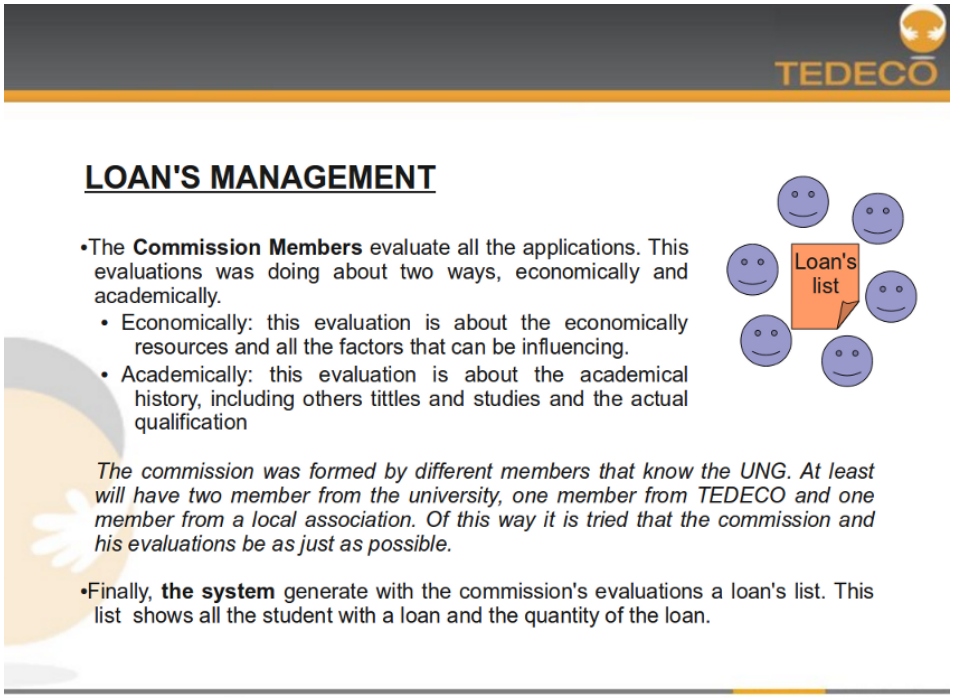


FIGURE 10.9. Pre-Deployment Presentation slide 9

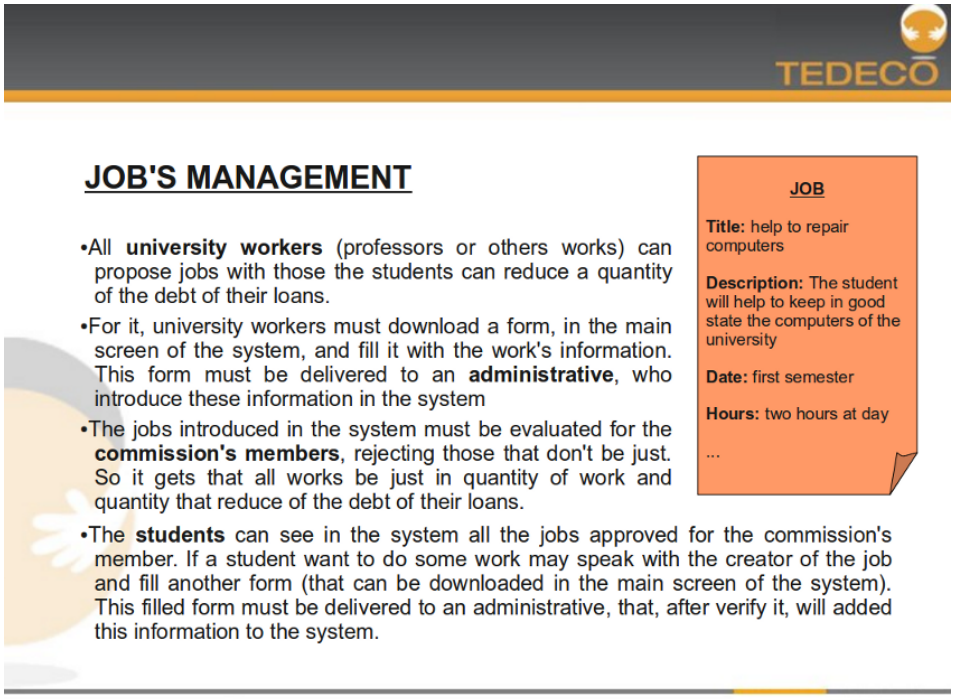


FIGURE 10.10. Pre-Deployment Presentation slide 10

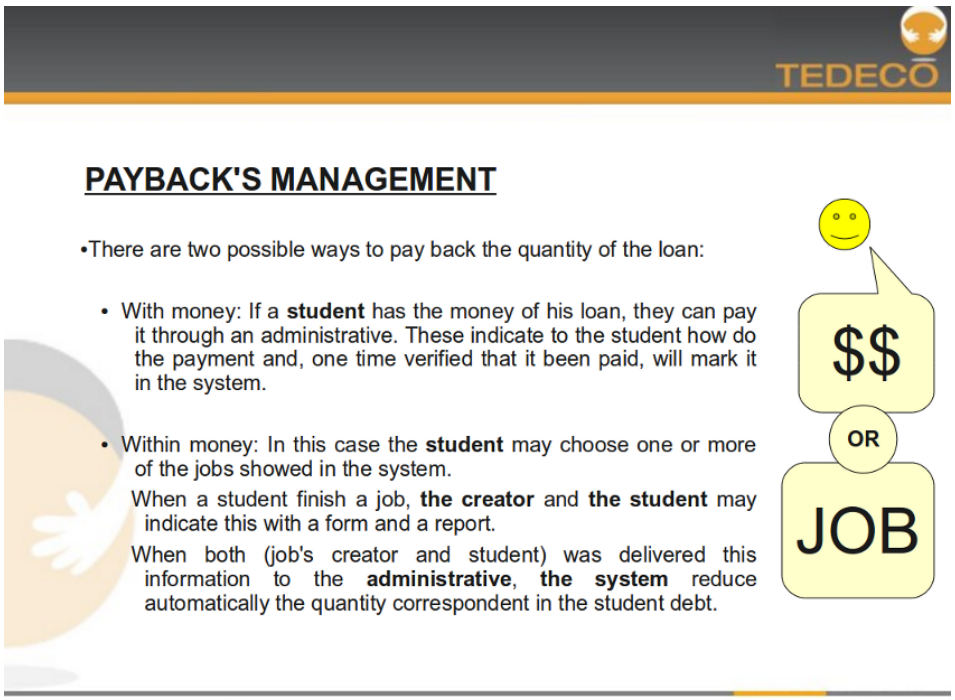


FIGURE 10.11. Pre-Deployment Presentation slide 11

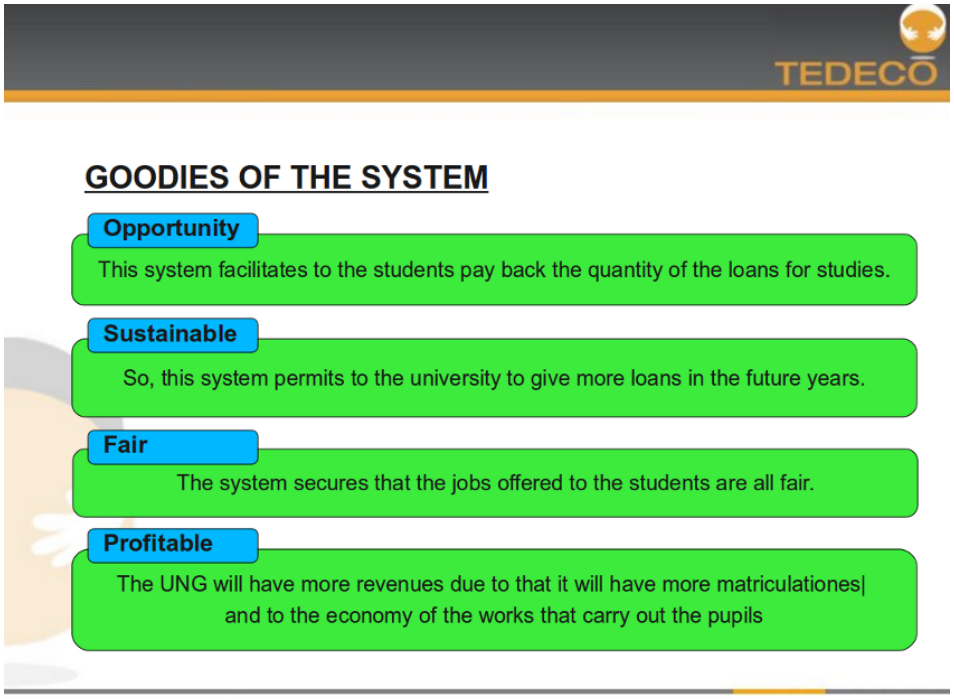


FIGURE 10.12. Pre-Deployment Presentation slide 12

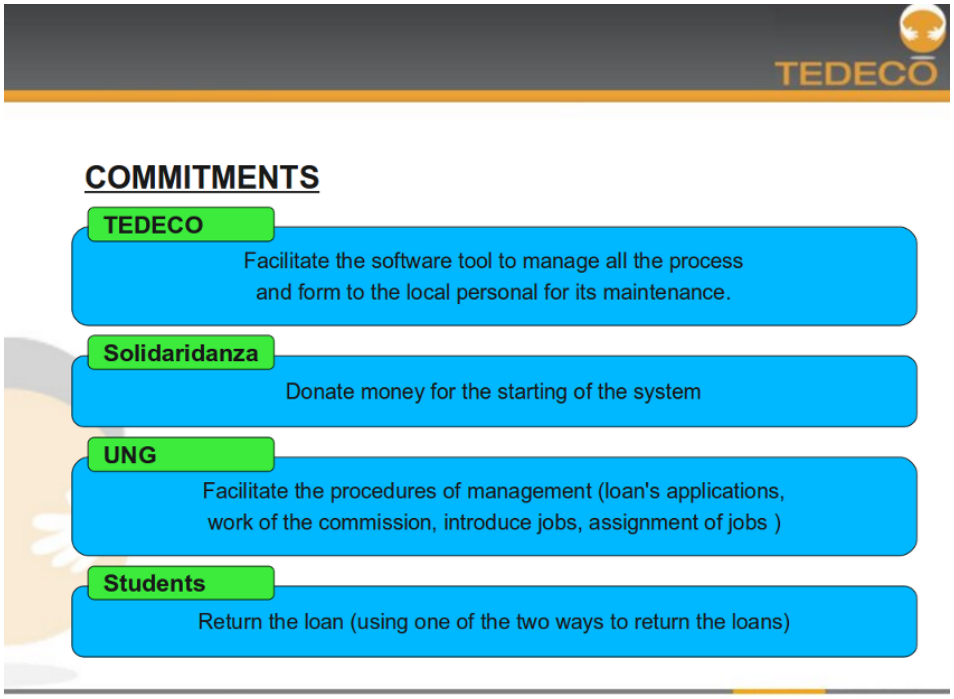


FIGURE 10.13. Pre-Deployment Presentation slide 13

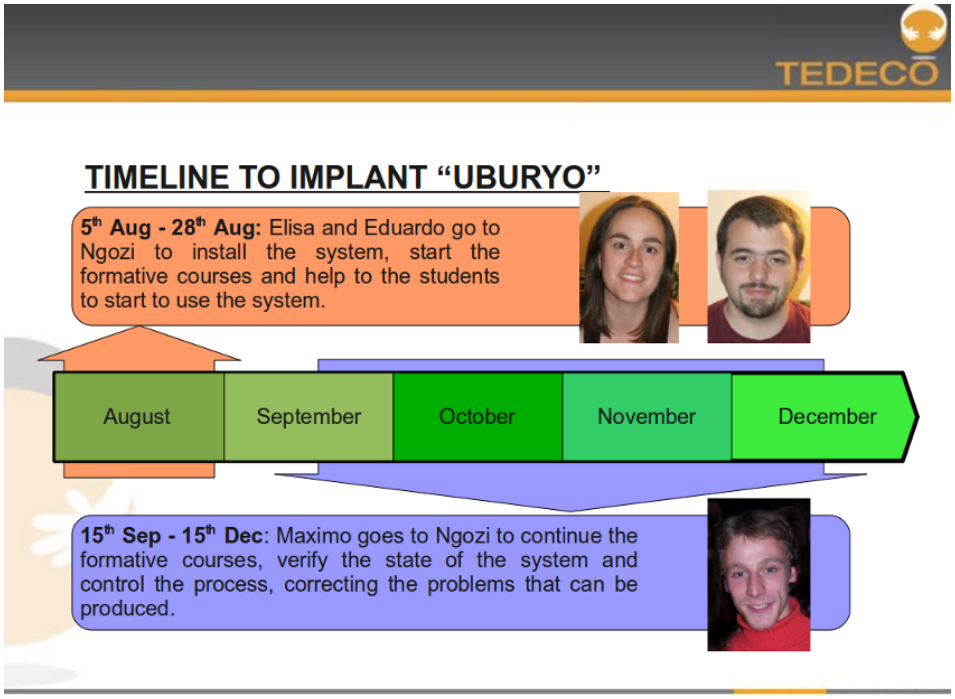


FIGURE 10.14. Pre-Deployment Presentation slide 14

FIGURE 10.15. Pre-Deployment Presentation slide 15

10.1.1.3. *SLIDE 3 [Figure 10.3]: THE PROBLEM OF LOANS TO STUDY. (animated).*

In this slide is explained a normal system of loans to study. Was showed how work in the theory *[is animated and now is explained following the clicks (each number is a different click)]*:

[At the beginning is showed the university with a loans bag and the students].

(1) First step: The university provides money to the students for realize the studies (leaving the bag empty)
(2) Second step: at the end of their studies and after introduce them in the working live, the students return the loan (fill up the bag to allow start the cycle one more time, allowing give loans to other students)

10.1.1.4. *SLIDE 4 [Figure 10.4]: THE PROBLEM OF LOANS TO STUDY. (animated).*

In this slide was showed how work in the reality (at the UNG) this normal system of loans to study. *[is animated and now is explained following the clicks (each number is a different click)]*:

[At the beginning is showed the university with a loans bag and the students].

(1) First step: The university provides money to the students for realize the studies (leaving the bag empty)
(2) Second step: The student, for many different problems, can not return the money of the loans (leaving the university with the bag empty). Then it is impossible to give loans in other time

10.1.1.5. *SLIDE 5 [Figure 10.5]: THE SUSTAINABLE SYSTEM OF LOANS. (animated).*

In this slide is showed the idea of a Sustainable System of Loans to study. *[is animated and now is explained following the clicks (each number is a different click)]*:

[At the beginning is showed the university with a loans bag and the students].

(1) First step: The university provides money to the students for realize the studies (leaving the bag empty)
(2) Second step: Some students return the loan with money (recovering immediately part of the money from the loans bag). Other students return the loan with works for the university *[At this point the people think that it is impossible to recover the money following this way]*
(3) Third step: The money do not recover in the last step was recover with the money that the university save when have some student doing some works.

10.1.1.6. *SLIDE 6 [Figure 10.6]: HOW WORK THIS SYSTEM? (animated).*

In this slide is showed with a graphic schema how work, in general terms, the system. *[is animated and now is explained following the clicks (each number is a different click)]*:

[At the beginning was showed a cube that represents the system].

(1) The students fill up the applications in the system. *[automatically after the click appear the student , the corresponding arrow and the applications]*
(2) The administrative validate the applications thanks to the extra documentation provided by the students *[after the click appear the administrative and the corresponding arrow]*
(3) The commission of loans evaluate the validated applications *[after the click appear the commission and the corresponding arrow]*

(4) The commission generates a list of loans, with the students that have a loan. Really was generated automatically by the system, but thanks to the ratings of the commission members.*[after the click appear the corresponding arrow and a list of loans]*
(5) The teachers and university workers of the university propose jobs so that students can return their loans. *[automatically after the click appear the university workers, the arrow and the works.]*
(6) The commission evaluates the different proposed jobs too. *[after the click appear the corresponding arrow]*
(7) The students with loans choose a job and do it. *[after the click appear the corresponding arrow]*

10.1.1.7. *SLIDE 7 [Figure 10.7]: HOW WORK THIS SYSTEM?(2/2).*

This slide shows the time line for a normal course or a normal announcement, using this Sustainable System of Loans. All the time periods and tentative dates and can be configured in the system.

10.1.1.8. *SLIDE 8 [Figure 10.8]: APPLICATION'S MANAGEMENT..*

[At this point start a self-contained slide group, that shows in detail how run the principal tasks of the system, following the different roles in this system].

This slide shows hos is the application management (very important for the students) If have date for the implantation is the moment to say it..

10.1.1.9. *SLIDE 9 [Figure 10.9]: LOAN'S MANAGEMENT..*

This slide shows how is the loans management (very important for the commission member).

10.1.1.10. *SLIDE 10 [Figure 10.10]: JOB'S MANAGEMENT..*

This slide shows how is the job management (very important for the university workers and the professors).

10.1.1.11. *SLIDE 11 [Figure 10.11]: PAY BACK'S MANAGEMENT..*

This slide shows how is the return of loans managements (very important for the students and the university workers and the professors).

10.1.1.12. *SLIDE 12 [Figure 10.12]: GOODIES OF THE SYSTEM. (animated).*

This slide shows four main positive keys of the system, in order to summarize after the talk. *[With each click (4) appear each key].*

10.1.1.13. *SLIDE 13 [Figure 10.13]: COMMITMENTS. (animated).*

This slide shows four commitments that is important do refer to the system. Can be a problematic point and can be eliminated from the presentation *[With each click (4) appear each commitment].*

10.1.1.14. *SLIDE 14 [Figure 10.14]: TIMELINE TO IMPLANT "UBURYO". (animated).*

This slide shows the time line for the deployment of "Uburyo", so the people that realize this deployment. *[At the beginning appear the months between August and December (each number is a different click)].*

(1) Appear an arrow and the first step, with the corresponding tasks and how is going to do them.
(2) Appear a second arrow and the second step, with the corresponding tasks and how is going to do them.

10.1.1.15. *SLIDE 15 [Figure 10.15]: QUESTIONS..*

No comment :-D.

10.2. Uburyo Presentation for Solidaridanza

FIGURE 10.16. Solidaridanza Presentation slide 1

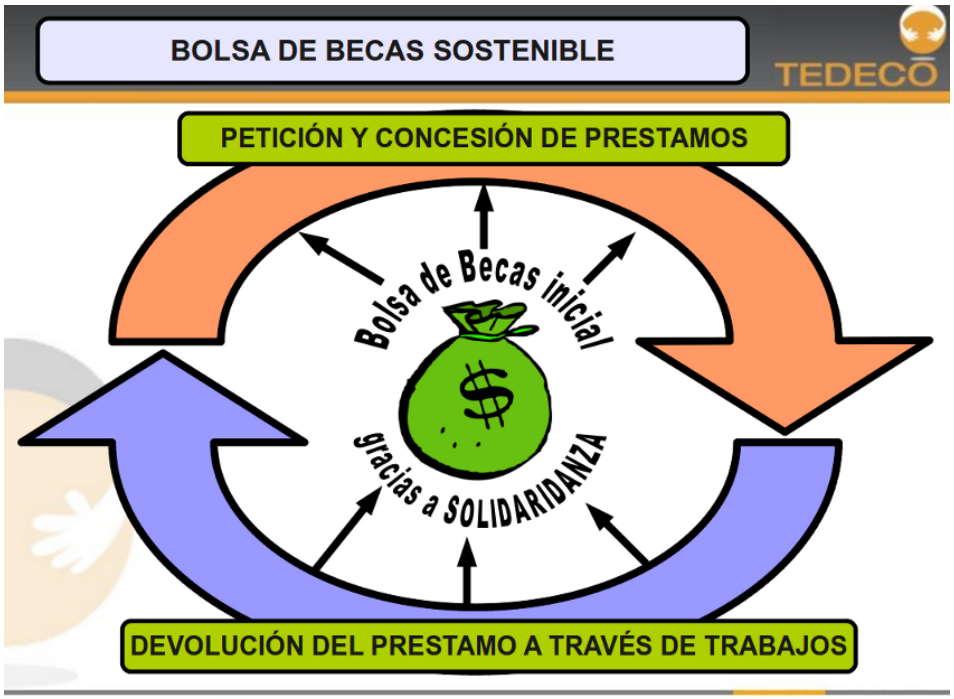


FIGURE 10.17. Solidaridanza Presentation slide 2

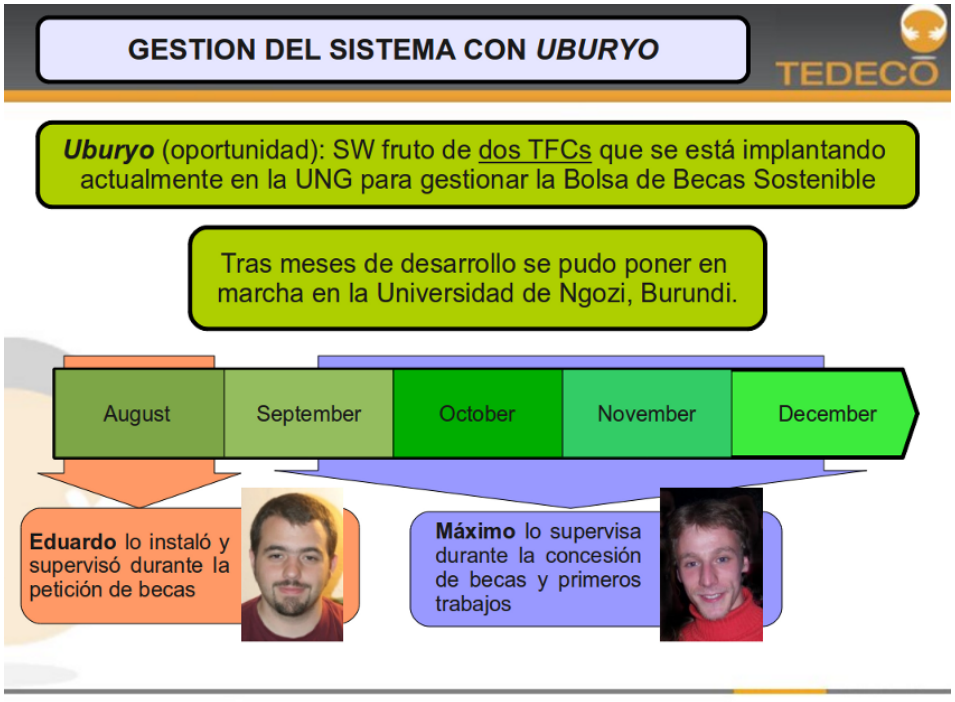


FIGURE 10.18. Solidaridanza Presentation slide 3

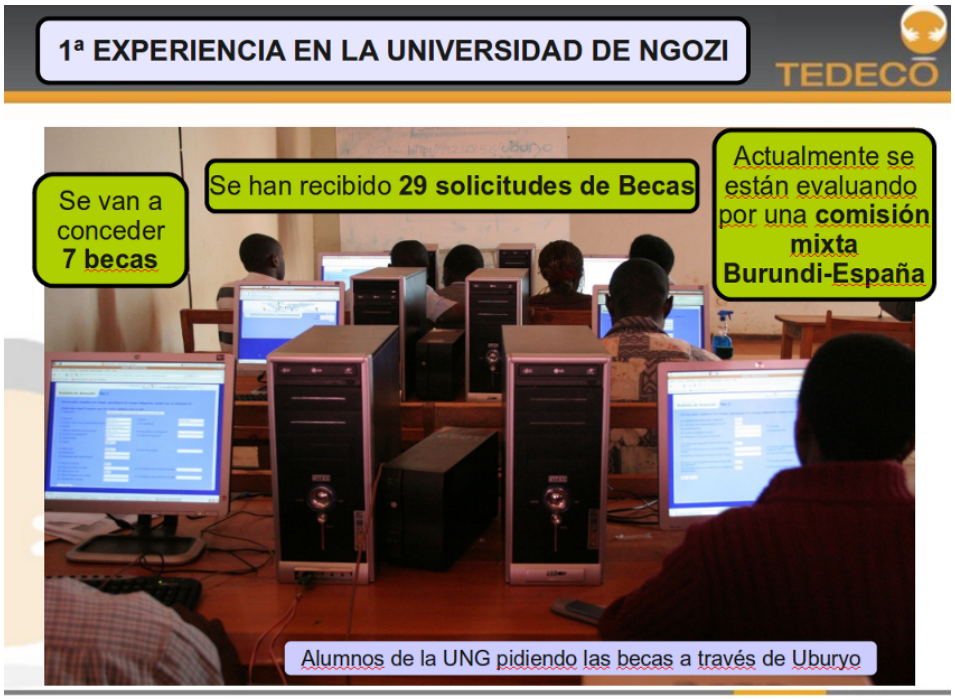


FIGURE 10.19. Solidaridanza Presentation slide 4

This presentation was created for the association Solidaridanza [SOL10], more specifically for the 2009 annual dance meeting. It is important remember that Solidaridanza provided the money for the first scholarship bag managed by Uburyo. This presentations [Figures 10.16, 10.17, 10.18 y 10.19] was written in Spanish.

10.3. Poster for III Inter. Meeting on ICT for Development Coop.

The following poster [Figure 10.20] was created to present the project in the III International Meeting on ICT for Development Cooperation (III Coop2.0) [COO10]. This meeting took place at LABoral Centro de Arte y Creación Industrial - (Gijón - Spain) at 1st – 2nd March 2010. With the image there is a descriptive text that accompanied the poster. Both, poster and text, was written in Spanish.

10.3.1. Uburyo poster summary.

Uburyo: Sistema de Becas Sostenible.

Resumen del póster.

Para la mayoría de la población de países en vías de desarrollo, el acceso a estudios superiores sigue siendo un problema por las escasas plazas estatales y el precio de la matricula en los estudios privados. Una medida paliativa a este problema es la de la asignación de becas/préstamo para que los alumnos puedan sufragar sus gastos de matriculación. Sin embargo, los sistemas de becas tradicionales se encuentran con numerosos problemas para su sostenibilidad: la devolución del importe de las becas es en raras ocasiones restituido para su posterior asignación, existen numerosos casos de corrupción en la asignación de las becas y existe una dependencia de financiación externa para su continuidad.

Desde el grupo de cooperación TEDECO (Tecnologías para el Desarrollo y la Cooperación) de la UPM se está trabajando en un sistema sostenible de préstamos para el estudio y en Uburyo ("oportunidad" en lengua Kirundi), una aplicación informática que lo gestione.

Todo el proyecto está especialmente destinado a centros de educación superior en países en vías de desarrollo, aunque podría ser replicado y adaptado a otros tipos de centros educativos.

La idea básica es que la institución educativa mantenga una bolsa de becas que son prestadas a los alumnos que mas lo necesiten o merezcan. Tras disfrutarlas, los alumnos que han recibido dichas becas deberán devolverla a la universidad para que está pueda seguir prestándolas a los alumnos futuros. Para ello se ofrece la posibilidad de devolver el préstamo realizando diversos trabajos para la institución. De esta manera la institución retorna el valor de las becas ahorrando en el pago de estos trabajos, o a través de cierto valor añadido, como sería conseguir más alumnos al mejorar las instalaciones.

Aunque es pronto para sacar conclusiones, ya que aún se está realizando la primera experiencia piloto, podemos decir que el sistema de préstamos no solo ayuda a los estudiantes sino que beneficia económicamente a las instituciones educativas. Además es destacable el cambio de mentalidad de los alumnos becados que se sienten responsables de devolver sus prestamos para no negar a otro estudiante que venga tras ellos la oportunidad que se les está dando a ellos.

FIGURE 10.20. Uburyo poster